**Table of Contents**

**Lab 5: Accessing an Oracle Database**

Practices for Lesson 5: Overview

Practice 5-1: Viewing Database Deployment Information and Menus

Practice 5-2: Connecting to the Database Deployment Compute Node

Practice 5-3: Exploring a CDB by Using SQL\*Plus

Practice 5-4: Exploring a PDB by Using SQL\*Plus

Practice 5-5: Installing the HR Sample Schema

Practice 5-6: Copying Course Practice Files

**Lab 6: Managing DBCS Database Deployments**

Practices for Lesson 6: Overview

Practice 6-1: Accessing Enterprise Manager Database Express

Practice 6-2: Exploring a CDB and PDB by Using Enterprise Manager Database Express

**Lab 7: Managing Database Instances**

Practices for Lesson 7: Overview

Practice 7-1: Investigating Initialization Parameter Files

Practice 7-2: Viewing Initialization Parameters by Using SQL\*Plus

Practice 7-3: Modifying Initialization Parameters by Using SQL\*Plus

Practice 7-4: Modifying Initialization Parameters by Using Enterprise Manager Database

Express

Practice 7-5: Shutting Down and Starting Up the Oracle Database

Practice 7-6: Viewing Diagnostic Information

**Lab 8: Understanding Oracle Net Services**

[Practices for Lesson 8: Overview](#_TOC_250031)

[Practice 8-1: Exploring the Default Listener](#_TOC_250030)

[Practice 8-2: Creating a Static Listener for a PDB](#_TOC_250029)

[Practice 8-3: Verifying the Net Service Name for PDB1](#_TOC_250028)

Practices for Lesson 9: Administering User Security

[Practices for Lesson 9: Overview](#_TOC_250027)

[Practice 9-1: Creating Common and Local Users](#_TOC_250026)

[Practice 9-2: Creating a Local User for an Application](#_TOC_250025)

[Practice 9-3: Granting a Local Role (DBA) to PDBADMIN](#_TOC_250024)

[Practice 9-4: Using EM Express to Create a Local Profile](#_TOC_250023)

[Practice 9-5: Using EM Express to Create Local Roles](#_TOC_250022)

[Practice 9-6: Using EM Express to Create Local Users](#_TOC_250021)

[Practice 9-7: Configuring a Default Role for a User](#_TOC_250020)

[Practice 9-8: Exploring OS and Password File Authentication](#_TOC_250019)

Lab 10: Creating PDBs

[Practices for Lesson 10: Overview](#_TOC_250018)

[Practice 10-1: Creating a PDB from Seed](#_TOC_250017)

[Practice 10-2: Cloning a PDB](#_TOC_250016)

[Practice 10-3: Unplugging and Plugging in a PDB](#_TOC_250015)

[Practice 10-4: Dropping a PDB](#_TOC_250014)

[Lab 11: Creating Master Encryption Keys for PDBs](#_TOC_250013)

[Practices for Lesson 11: Overview](#_TOC_250012)

[Practice 11-1: Creating and Activating an Encryption Key](#_TOC_250011)

[Practice 11-2: Creating and Activating the Encryption Key for PDB2](#_TOC_250010)

Lab 12: Creating and Managing Tablespaces

[Practices for Lesson 12: Overview](#_TOC_250009)

[Practice 12-1: Viewing Tablespace Information](#_TOC_250008)

[Practice 12-2: Creating a Tablespace](#_TOC_250007)

[Practice 12-3: Creating a Tablespace that is Encrypted by Default](#_TOC_250006)

Lab 13: Managing Storage Space

[Practices for Lesson 13: Overview](#_TOC_250005)

[Practice 13-1: Managing Space in Tablespaces](#_TOC_250004)

[Practice 13-2: Using Compression](#_TOC_250003)

[Practice 13-3: Enabling the Resumable Space Allocation Feature](#_TOC_250002)

Lab 14: Managing Undo Data

[Practices for Lesson 14: Overview](#_TOC_250001)

[Practice 14-1: Managing Undo Data](#_TOC_250000)

Lab 15: Moving Data

Practices for Lesson 15: Overview

Practice 15-1: Moving Data from One PDB to Another PDB

Practice 15-2: Loading Data into a PDB from an External File

**Lab 16: Backup and Recovery Concepts**

Practices for Lesson 16

**Lab 17: Backup and Recovery Configuration**

Practices for Lesson 17: Overview

Practice 17-1: Verifying that the Control File is Multiplexed

Practice 17-2: Checking Storage Availability

Practice 17-3: Configuring the Size of the Fast Recovery Area

Practice 17-4: Verifying that the Redo Log File is Multiplexed

Practice 17-5: Verifying that ARCHIVELOG Mode is Configured

**Lab 18: Creating Database Backups**

Practices for Lesson 18: Overview

Practice 18-1: Backing up the Control File

Practice 18-2: Verifying Automatic Backups of the Control File and SPFILE

Practice 18-3: Checking Storage Availability

Practice 18-4: Creating a Whole Database Backup

Practice 18-6: Creating Partial Database Backups

**Lab 19: Performing Database Recovery**

Practices for Lesson 19: Overview

Practice 19-1: Recovering from the Loss of a System-Critical Data File

Practice 19-2: Recovering from the Loss of an Application Data File

**Practices for Lesson 20: Monitoring and Tuning Database Performance**

Practices for Lesson 20: Overview

Practice 20-1: Using Enterprise Manager Database Express to Manage Performance

Practice 20-2: Resolving Lock Conflicts

**Lab 21: SQL Tuning**

Practices for Lesson 21: Overview

Practice 21-1: Using the SQL Tuning Advisor

Practice 21-2: Using the Optimizer Statistics Advisor

**Practice 5-1: Start Oracle Database**

### Overview

In this practice, you will start oracle database which has been already created as follows:

ORACLE\_SID=fenagodb

ORACLE\_PDB=fenagodb1

ORACLE\_PWD=fenago

### Tasks

1. Log in to your lab environment and double click “Start Oracle Database” shortcut.



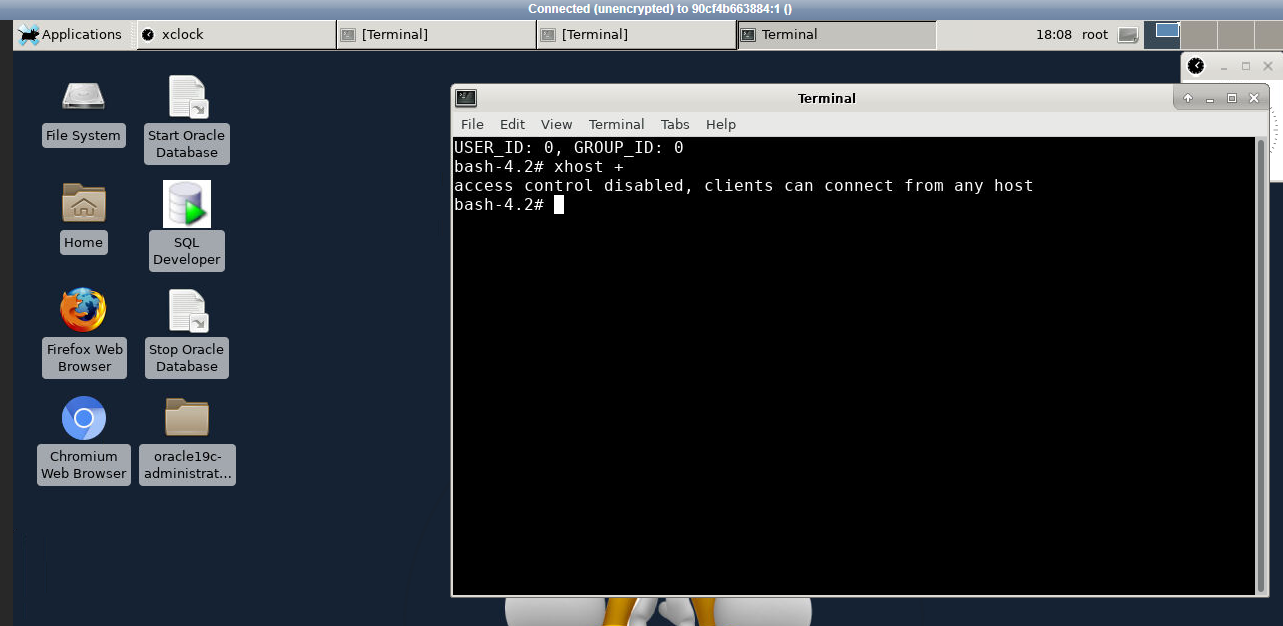
**Practice 5-2: Switch to oracle user from terminal**

### Overview

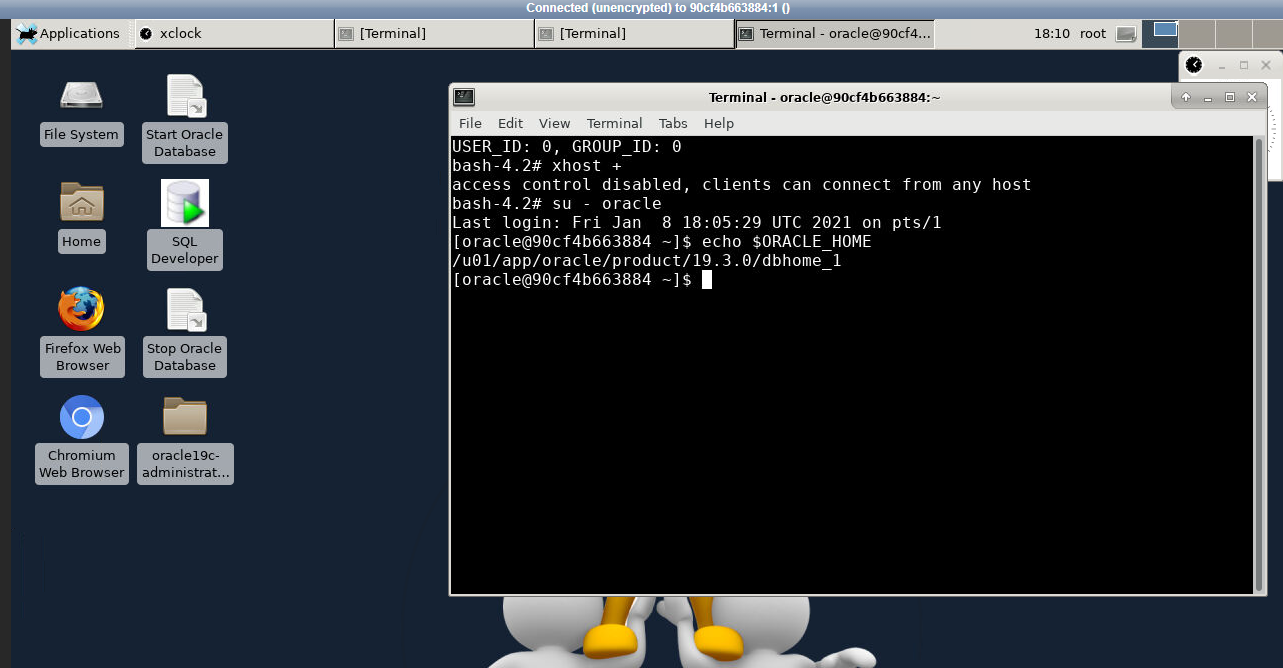
In this practice, you will switch to oracle user from terminal

### Tasks

* 1. Open terminal and run “xhost +” command as root user:



b. Run and run “su - oracle” command in the terminal to switch to **oracle** user:



**Practice 5-3: Exploring a CDB by Using SQL\*Plus**

### Overview

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

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

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

* Log in to the root container as a common user. A CDB common user is a database account created in the root container and is inherited by all PDBs in the CDB.
* Query container data objects, such as views whose names begin with V$ and CDB\_. For more information, refer to the following sections in *Oracle Database Administrator's Guide*:
* About Viewing Information When the Current Container is the CDB Root
* Viewing Information About the Containers in a CDB

In some of the steps below, you will format columns by using the COLUMN command. For example, applying the format A55 specifies an alphabetic format of 55 characters wide. Format 999 is an example of a numeric format.

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

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

### Assumptions

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

### Tasks

1. Set the Oracle environment variables. You need to set these each time you open a new terminal window.
   1. In the terminal window, list the search path that holds the oraenv script.

[oracle@MYDBCS ~]$ **which oraenv**

/u01/app/oracle/product/19.3.0/dbhome\_1/bin/oraenv

[oracle@MYDBCS ~]$

* 1. Source the oraenv script. oraenv sets the required environment variables needed for you to connect to your database instance. The oraenv script sets the ORACLE\_SID and ORACLE\_HOME environment variables and includes the $ORACLE\_HOME/bin directory in the PATH environment variable setting. Environment variables that this

script sets will persist in the terminal window until you close it. For the ORACLE\_SID

value, enter FENAGODB.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [FENAGODB] ? **FENAGODB**

The Oracle base has been set to /u01/app/oracle [oracle@MYDBCS ~]$

* 1. View the environment variables set by the oraenv command.

[oracle@MYDBCS ~]$ **set | grep ORACLE** OLD\_ORACLE\_BASE=/u01/app/oracle ORACLE\_BASE=/u01/app/oracle ORACLE\_HOME=/u01/app/oracle/product/19.3.0/dbhome\_1

ORACLE\_HOSTNAME=MYDBCS.compute-588436052.oraclecloud.internal ORACLE\_SID=FENAGODB

ORACLE\_UNQNAME=FENAGODB

[oracle@MYDBCS ~]$

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

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

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

SQL\*Plus: Release 18.0.0.0.0 Production on Tue May 29 20:18:18

2018

Version 19.3.0.0.0

Copyright (c) 1982, 2017, Oracle. All rights reserved.

Connected to:

Oracle Database 19c EE High Perf Release 18.0.0.0.0 - Production Version 19.3.0.0.0

SQL>

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

SQL> **SHOW user**

USER is "SYS"

SQL>

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

--------- --- ---------- FENAGODB YES 0

SQL>

CON\_ID

CDB

NAME

SQL> **SELECT name, cdb, con\_id FROM v$database;**

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

SQL> **SHOW con\_name**

CON\_NAME

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

SQL>

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

SQL> **SHOW con\_id**

CON\_ID

------------------------------ 1

SQL>

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

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

BANNER

----------------------------------------------------------------

Oracle Database 19c EE High Perf Release 18.0.0.0.0 - Production

SQL>

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

-------- ---------- CDB$ROOT 1

PDB$SEED 2

PDB1 3

SQL>

CON\_ID

NAME

SQL> **COLUMN name FORMAT A8**

SQL> **SELECT name, con\_id FROM v$containers ORDER BY con\_id;**

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

SQL>

READ WRITE NO

NO

READ ONLY

1. PDB$SEED
2. PDB1

---------- ------------------------------ ---------- ----------

OPEN MODE RESTRICTED

SQL> **SHOW pdbs**

CON\_ID CON\_NAME

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

SQL> **COLUMN pdb\_name FORMAT A8**

SQL> **SELECT pdb\_name, status FROM cdb\_pdbs ORDER BY 1;**

PDB\_NAME STATUS

-------- ---------- PDB1 NORMAL PDB$SEED NORMAL

SQL>

1. View information about the data files in your CDB.
   1. List all the data files in the CDB (for the root container and all PDBs) by querying the

CDB\_DATA\_FILES view. The order of your results may vary.

SQL> **COLUMN file\_name FORMAT A50**

SQL> **COLUMN tablespace\_name FORMAT A10**

SQL> **SELECT file\_name, tablespace\_name FROM cdb\_data\_files;**

FILE\_NAME TABLESPACE

-------------------------------------------------- ----------

/u02/app/oracle/oradata/FENAGODB/users01.dbf USERS

/u02/app/oracle/oradata/FENAGODB/undotbs01.dbf UNDOTBS1

/u02/app/oracle/oradata/FENAGODB/system01.dbf SYSTEM

/u02/app/oracle/oradata/FENAGODB/sysaux01.dbf SYSAUX

/u02/app/oracle/oradata/FENAGODB/PDB1/system01.dbf SYSTEM

/u02/app/oracle/oradata/FENAGODB/PDB1/sysaux01.dbf SYSAUX

/u02/app/oracle/oradata/FENAGODB/PDB1/undotbs01.dbf UNDOTBS1

/u02/app/oracle/oradata/FENAGODB/PDB1/PDB1\_users01.dbf USERS

8 rows selected.

SQL>

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

SQL> **COL name FORMAT A12**

SQL> **SELECT d.file#, ts.name, ts.ts#, ts.con\_id**

1. **FROM v$datafile d, v$tablespace ts**
2. **WHERE d.ts#=ts.ts# AND d.con\_id=ts.con\_id**
3. **ORDER BY 4;**

12 rows selected.

SQL>

|  |  |  |  |
| --- | --- | --- | --- |
| FILE# | NAME | TS# | CON\_ID |
| ---------- | ------------ | ---------- | ---------- |
| 1 | SYSTEM | 0 | 1 |
| 3 | SYSAUX | 1 | 1 |
| 4 | UNDOTBS1 | 2 | 1 |
| 7 | USERS | 4 | 1 |
| 6 | SYSAUX | 1 | 2 |
| 13 | USERS | 5 | 2 |
| 8 | UNDOTBS1 | 2 | 2 |
| 5 | SYSTEM | 0 | 2 |
| 9 | SYSTEM | 0 | 3 |
| 10 | SYSAUX | 1 | 3 |
| 11 | UNDOTBS1 | 2 | 3 |
| 12 | USERS | 5 | 3 |

* 1. List all temp files in the CDB (for the root container and all PDBs) by querying the

CDB\_TEMP\_FILES view.

SQL> **SELECT file\_name, tablespace\_name FROM cdb\_temp\_files;**

FILE\_NAME TABLESPACE

-------------------------------------------------- ----------

/u04/app/oracle/oradata/temp/temp01.dbf TEMP

/u02/app/oracle/oradata/FENAGODB/PDB1/pdbseed\_temp0120 TEMP 18-02-19\_18-48-12-642-PM.dbf

SQL>

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

---------- ------------------------------------------ ----------

3 /u04/app/oracle/redo/redo03.log 0

2 /u04/app/oracle/redo/redo02.log 0

1 /u04/app/oracle/redo/redo01.log 0

SQL>

CON\_ID

SQL> **COLUMN member FORMAT A42**

SQL> **SELECT group#, member, con\_id FROM v$logfile;**

GROUP# MEMBER

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

------------------------------------------------------- --------

/u02/app/oracle/oradata/FENAGODB/control01.ctl 0

/u03/app/oracle/fast\_recovery\_area/FENAGODB/control02.ctl 0

SQL>

CON\_ID

SQL> **COLUMN name FORMAT A55**

SQL> **SELECT name, con\_id FROM v$controlfile;**

NAME

1. View information about the pre-created users in your CDB.
   1. List only the common users in the CDB by querying the CDB\_USERS view.

SQL> **SELECT DISTINCT username FROM cdb\_users**

2 **WHERE common ='YES' ORDER BY 1;**

USERNAME

----------------------------------------------------------------

ANONYMOUS APPQOSSYS AUDSYS C##DBAAS\_BACKUP

… SYSTEM WMSYS XDB XS$NULL

38 rows selected.

SQL>

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

SQL> **COLUMN username FORMAT A25**

SQL> **SELECT con\_id, username FROM cdb\_users**

2 **ORDER BY username, con\_id;**

CON\_ID USERNAME

---------- -------------------------

1 ANONYMOUS

3 ANONYMOUS 3 APEX\_050100

3 APEX\_INSTANCE\_ADMIN\_USER

…

1 OUTLN

3 OUTLN

3 PDBADMIN

1 REMOTE\_SCHEDULER\_AGENT

3 REMOTE\_SCHEDULER\_AGENT

…

1 SYS

3 SYS

…

1 SYSRAC

3 SYSRAC

1 SYSTEM

3 SYSTEM

…

84 rows selected.

SQL>

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

---------------- ------------ ---------- FENAGODB OPEN 0

SQL>

CON\_ID

STATUS

INSTANCE\_NAME

SQL> **SELECT instance\_name, status, con\_id FROM v$instance;**

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

SQL> **SELECT con\_id, name FROM v$services ORDER BY 1;**

CON\_ID NAME

---------- -----------------------------------------------------

1 SYS$BACKGROUND

1 FENAGODB.588436052.oraclecloud.internal

1 FENAGODB.588436052.oraclecloud.internalXDB

1 SYS$USERS

3 pdb1

SQL>

1. Exit SQL\*Plus.

SQL > **exit**

Disconnected from Oracle Database 19c EE High Perf Release

18.0.0.0.0 - Production Version 19.3.0.0.0

[oracle@MYDBCS ~]$

**Practice 5-4: Exploring a PDB by Using SQL\*Plus**

### Overview

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

* Connect to a PDB indirectly through a CDB
* Query the data dictionary to view information about data files, temp files, and users in a PDB
* Connect to a PDB directly by using the Easy Connect syntax To find data dictionary information specific to a root container or a PDB:
* Query DBA\_ views to return container-specific information.
* When you are logged in to a PDB, queries against the data dictionary return information about that PDB only, regardless of the view you query.
* When queried from a PDB, the DBA\_PDBS view returns the information related to the PDB to which you are connected. When queried from the root container, the DBA\_PDBS view provides information on all PDBs belonging to a given CDB.

### Assumptions

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

oracle user.

### Tasks

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

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

… SQL>

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

OPEN\_MODE

CON\_ID NAME

SQL> **COLUMN con\_id FORMAT 999**

SQL> **COLUMN name FORMAT A10**

SQL> **SELECT con\_id, name, open\_mode FROM v$pdbs;**

------ ---------- ----------

1. PDB$SEED READ ONLY
2. PDB1 READ WRITE

SQL>

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

SQL> **ALTER PLUGGABLE DATABASE PDB1 OPEN;**

Pluggable database altered. SQL>

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

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

Session altered.

SQL>

* 1. Verify that the container name is PDB1.

SQL> **SHOW con\_name**

CON\_NAME

------------------------------ PDB1

SQL>

1. Query the data dictionary to list the data files and temp files for PDB1.
   1. List the data files for PDB1 and the tablespaces to which they belong by querying the

DBA\_DATA\_FILES view.

----------------------------------------------------- ----------

/u02/app/oracle/oradata/FENAGODB/PDB1/system01.dbf SYSTEM

/u02/app/oracle/oradata/FENAGODB/PDB1/sysaux01.dbf SYSAUX

/u02/app/oracle/oradata/FENAGODB/PDB1/undotbs01.dbf UNDOTBS1

TABLESPACE

FILE\_NAME

SQL> **col file\_name format a60**

SQL> **col tablespace\_name format a10**

SQL> **SELECT file\_name, tablespace\_name FROM dba\_data\_files;**

SQL>

USERS

/u02/app/oracle/oradata/FENAGODB/PDB1/PDB1\_users01.dbf

* 1. List the temp files for PDB1 and the tablespaces to which they belong by querying the

DBA\_TEMP\_FILES view.

/u02/app/oracle/oradata/FENAGODB/PDB1/pdbseed\_temp012018-02-19\_1 TEMP 8-48-12-642-PM.dbf

SQL>

TABLESPACE

----------

FILE\_NAME

-----------------------------------------------------

SQL> **SELECT file\_name, tablespace\_name FROM dba\_temp\_files;**

* 1. List the local users for PDB1 by querying the DBA\_USERS view.

SQL> **SELECT DISTINCT username FROM dba\_users WHERE common='NO';**

USERNAME

---------------------------------------------------------------- PDBADMIN

APEX\_LISTENER

APEX\_PUBLIC\_USER APEX\_REST\_PUBLIC\_USER FLOWS\_FILES APEX\_050100

APEX\_INSTANCE\_ADMIN\_USER

SCOTT

8 rows selected.

SQL>

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

SQL > **DISCONNECT**

Disconnected from Oracle Database 19c EE High Perf Release

18.0.0.0.0 - Production Version 19.3.0.0.0

SQL>

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

SQL> **SHOW user**

USER is ""

SQL>

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

SQL> **CONNECT**

**system/*password*@localhost:1521/pdb1.588436052.oraclecloud.intern al**

Connected.

SQL>

* 1. Verify that you are now connected as the SYSTEM user by using the SHOW USER

command again.

SQL> **SHOW user**

SQL> USER is "SYSTEM"

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

**Practice 5-5: Installing the HR Sample Schema**

### Overview

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

### Assumptions

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

oracle user.

### Tasks

1. In your terminal window, navigate to the

$ORACLE\_HOME/demo/schema/human\_resources directory.

[oracle@MYDBCS ~]$ **cd $ORACLE\_HOME/demo/schema/human\_resources**

[oracle@MYDBCS human\_resources]$

1. Use the ls command to view the contents of the human\_resources directory. In a later step, you will execute the hr\_main.sql to create the HR user, objects and load data into the HR tables.

|  |  |  |  |
| --- | --- | --- | --- |
| [oracle@MYDBCS human\_resour  hr\_analz.sql hr\_comnt.sql hr\_main.sql | | ces]$ **ls**  hr\_drop\_new.sql | hr\_idx.sql |
| hr\_code.sql hr\_popul.sql | hr\_cre.sql | hr\_drop.sql | hr\_main\_new.sql |
| [oracle@MYDBCS human\_resources]$ | | | |

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

privilege.

[oracle@MYDBCS human\_resources]$ **sqlplus / as sysdba**

…

SQL>

1. Switch to PDB1.

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

Session altered.

SQL>

1. Execute the **hr\_main.sql** script and respond to the prompts as follows.
   1. Enter the password for the HR user as specified in the *Course Practice Environment: Security Credentials*.
   2. Enter **USERS** as the default tablespace for the HR user.
   3. Enter **TEMP** as the temporary tablespace for the HR user.
   4. Enter **$ORACLE\_HOME/demo/schema/log/** for the log directory.

SQL> **@hr\_main**

specify password for HR as parameter 1:

Enter value for 1: ***password***

specify default tablespeace for HR as parameter 2:

Enter value for 2: **USERS**

specify temporary tablespace for HR as parameter 3:

Enter value for 3: **TEMP**

specify log path as parameter 4:

Enter value for 4: **$ORACLE\_HOME/demo/schema/log/**

PL/SQL procedure successfully completed. User created.

User altered.

Grant succeeded.

…

Comment created.

Commit complete.

PL/SQL procedure successfully completed.

SQL>

* 1. Exit from SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS human\_resources]$

1. Query the USER\_TABLES view as the HR user to verify that the user and tables were created.
   1. Connect as the HR user. Be sure to provide the correct service name for your PDB as you did in Practice 5-4, step 3c.

[oracle@MYDBCS human\_resources]$ **sqlplus hr/*password*@localhost:1521/PDB1.588436052.oraclecloud.internal**

…

SQL>

* 1. Query USER\_TABLES.

SQL> **SELECT table\_name FROM user\_tables;**

TABLE\_NAME

---------------------------------------------------------------- REGIONS

COUNTRIES

LOCATIONS DEPARTMENTS JOBS EMPLOYEES JOB\_HISTORY

7 rows selected.

SQL>

1. Exit from SQL\*Plus and close the connection to the compute node.

SQL> **exit**

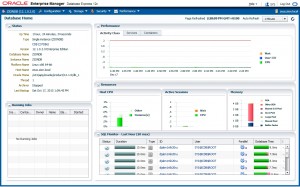
Disconnected from Oracle Database 19c Enterprise Edition Release

18.0.0.0.0 - Production Version 19.3.0.0.0

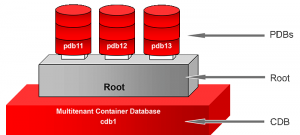
[oracle@MYDBCS human\_resources]$ **exit**

**How to enable Oracle Enterprise Manager Express 19c**

Oracle Enterprise Manager Express is a Web-based interface for managing an Oracle database 19c. It enables users to perform basic administrative tasks such as managing users, managing database initialization parameters, memory or storage. You can also view performance and SQL Tuning Advisor information, check status information about your database and pluggable databases.

**[](https://emarcel.com/wp-content/uploads/2015/12/oem_express_12c.jpg_01.jpg)**

The multi-tenant architecture enables an Oracle database to function as a multi-tenant container database (**CDB**) that includes zero, one, or many customer-created pluggable databases (**PDBs**).

[](https://emarcel.com/wp-content/uploads/2015/12/multitenant_db12c.png)

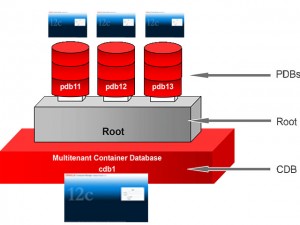
A **CDB** includes the following components:

**Root**named CDB$ROOT, stores Oracle-supplied metadata and common users. An example of metadata is the source code for Oracle-supplied PL/SQL packages. A common user is a database user known in every container.

A **PDB** appears to users and applications as if it were a non-CDB. For example, a PDB can contain the data and code required to support a specific application (e.g., APEX).

Each of these components is called a container. Therefore, the root is a container, the seed is a container, and each PDB is a container.

In this tutorial we will show two different types of configurations of Enterprise Manager Express one for CDB and the second for PDBs only. Imagine yourself as a dba who has full access to non-CDB/CDB/PDB,OEM Express 19c will allow you to manage CDB and all PDB containers from one central console. On the other hand you would like to allow regular users to login to OEM Express 19c as well, but grant them access to their PDBs only.

[](https://emarcel.com/wp-content/uploads/2015/12/multitenant_db12c_01.jpg)

In our demo environment we have the following containers created:

**CDB**: fenagodb,

**PDBs**: fenagodb1

**Configuring OEM Express for CDB (HTTPS)**

1. Open a terminal window, execute the “su – oracle” command to set the environment variables and connect to the multi-tenant container database (in our example **fenagodb**)Check if the database is a CDB database and it is open.

[root@fenago~]$ su – oracle

[oracle@fenago~]$ sqlplus / as sysdba

Connected to:

Oracle Database 19c Enterprise Edition Release 12.1.0.2.0 64bit

...

SQL> select name, cdb, con\_id from v$database;

NAME CDB CON\_ID

--------- --- ----------

FENAGODB YES 0

SQL> select instance\_name, status, con\_id from v$instance;

INSTANCE\_NAME STATUS CON\_ID

---------------- ------------ ----------

FENAGODB OPEN 0

2. Verify that the DISPATCHERS parameter in the initialization parameter file includes the PROTOCOL=TCP attribute.

SQL> show parameter dispatchers

NAME TYPE VALUE

------------------ ----------- ------------------------------

dispatchers string (PROTOCOL=TCP) (SERVICE=FENAGODBXDB)

max\_dispatchers integer

3. Execute the DBMS\_XDB.setHTTPSPort procedure to set the HTTPS port **5500**and the DBMS\_XDB.setHTTPPort procedure to set the HTTP port **5510**for EM Express

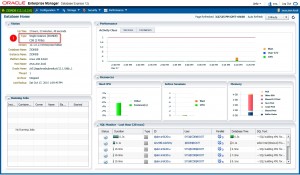
SQL> exec DBMS\_XDB\_CONFIG.SETHTTPSPORT(5500);

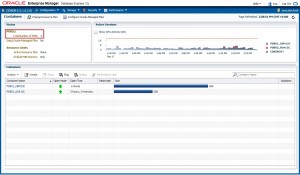
PL/SQL procedure successfully completed.

5. Login to Database EM Express Home Page.

**https:**//<IP:Hostname>:**5500**/

**Note:**Now we have the privileges to manage **CDB** and **PDBs**containers

**[](https://emarcel.com/wp-content/uploads/2015/12/CDB_oem_express_12c_01.jpg)**

**[](https://emarcel.com/wp-content/uploads/2015/12/CDB_oem_express_12c_02.jpg)**

**Configuring OEM Express for PDB Fenagodb1**

We are configuring EM Express for Fenagodb1 container to run on ports: HTTPS **5501** .

1. Display all pluggable databases and their status

SQL> select NAME, OPEN\_MODE from v$pdbs;

NAME OPEN\_MODE

------------------------------ ----------

PDB$SEED READ ONLY

FENAGODB1 READ WRITE

2. Alter the session and set container as **FENAGODB1**

SQL> alter session set container=FENAGODB1;

Session altered.

3. Execute the DBMS\_XDB.setHTTPSPort procedure to set the HTTPS port for EM Express

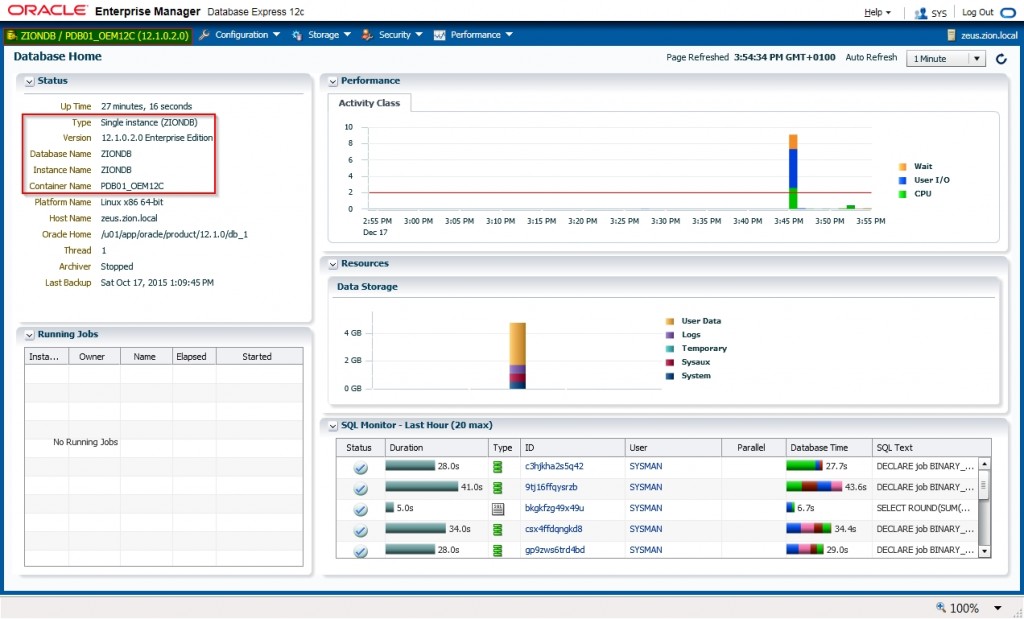
SQL> exec DBMS\_XDB\_CONFIG.SETHTTPSPORT(5501);

PL/SQL procedure successfully completed.

5. Login to Database EM Express home page.

**https:**//<IP:Hostname>:**5501**/em

Note that you have privileges to managed only PDB **FENAGODB1**

[](https://emarcel.com/wp-content/uploads/2015/12/EM-Express-12c-PDB.jpg)

Repeat recent steps from 2 to 5 to configure EM Express for more PDBs.

**Checking OEM Express port for CDB or PDB**

Alter session to CDB or PDB container and execute the SQL statement that returns the port that is configured for EM Express

SQL> select DBMS\_XDB\_CONFIG.GETHTTPSPORT() from dual;

DBMS\_XDB\_CONFIG.GETHTTPPORT()

----------------------------- 5501

If returned port number is 0, it means that EM Express is not configured for that particular container.

**Oracle Database 18c: Administration Workshop**

**Practices for Lesson 6: Overview**

### Overview

In these practices, you will use Enterprise Manager Database Express to explore your database.

**Practice 6-1: Accessing Enterprise Manager Database Express**

### Overview

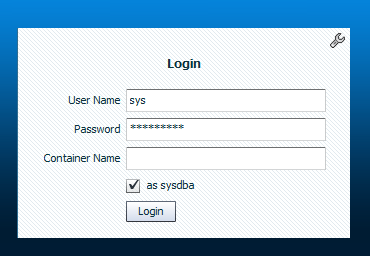
In this practice, you enable access to Enterprise Manager Database Express by creating an SSH tunnel for port forwarding.

### Assumptions

You successfully created a database deployment in a previous practice. You have access to PuTTY on a Windows system or the ssh utility on Linux.

### Tasks

1. Open a browser such as Firefox. Launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. If you receive the Firefox "Your connection is not secure" message, perform the following steps:
   1. Click **Advanced**.
   2. Click **Add Exception**.
   3. Click **Confirm Security Exception**.
3. On the Enterprise Manager Database Express login page, enter the following:
   1. In the User Name field, enter **SYS**.
   2. In the Password field, enter the password you specified when you created the database deployment.
   3. Leave the Container Name field empty.
   4. Click **Login**.



**Practice 6-2: Exploring a CDB and PDB by Using Enterprise Manager Database Express**

### Overview

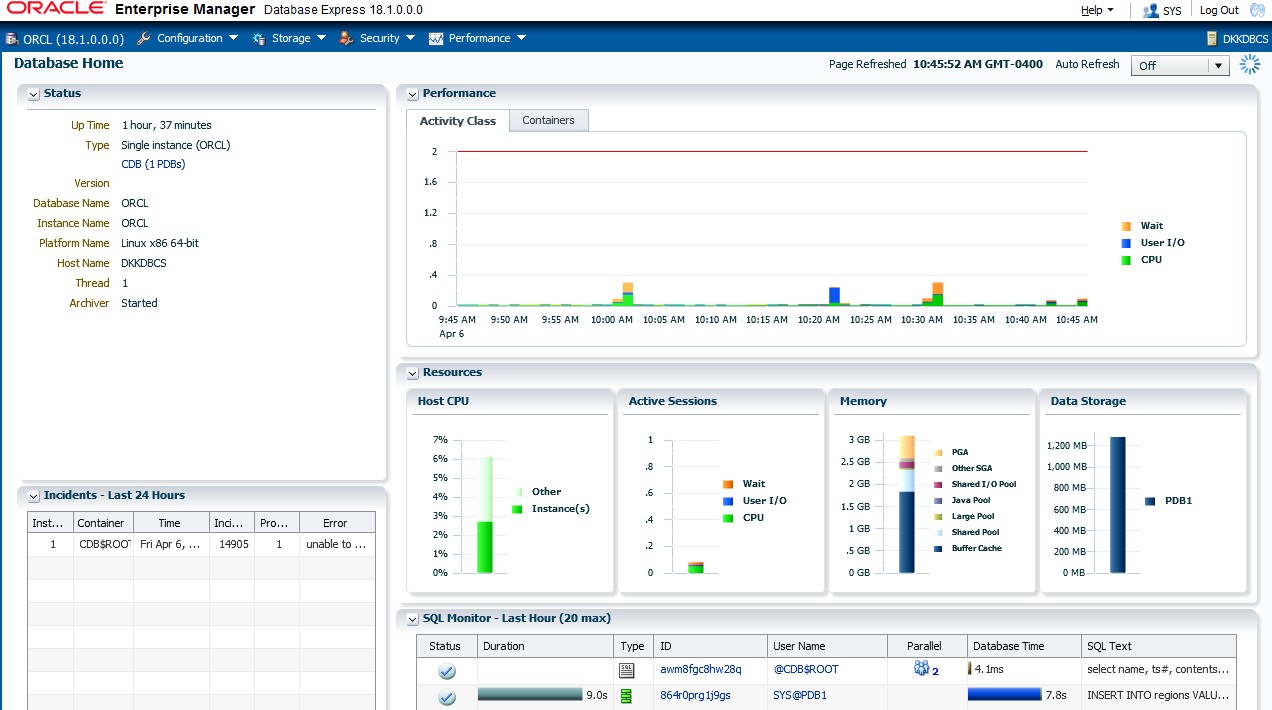
In this practice, you explore the differences in the Enterprise Manager Database Express interface when connected to a CDB and a PDB.

### Assumptions

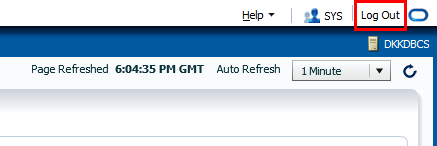
You logged in to Enterprise Manager Database Express as the SYS user in the previous practice.

### Tasks

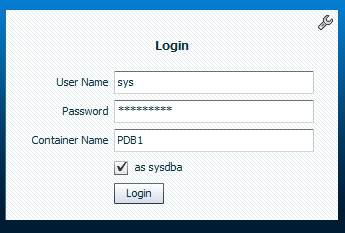
1. View the EM Express Home page for the CDB.



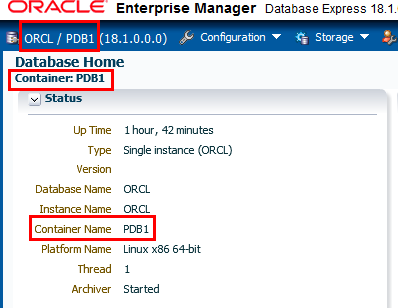
1. Click the tabs and menus to review the interface.
2. In the upper right corner, click **Log Out**.



1. Now log in to EM Express to view information for PDB1.
   1. On the Login page for EM Express, in the User Name field enter **SYS**.
   2. In the Password field, enter the password specified when you created the database deployment.
   3. Click **Login**.



1. View the EM Express Home page for PDB1. Notice that in the upper left corner, the Home page identifies the container. In the Status area, the Container Name field also indicates the container.



1. Browse the Configuration, Storage, Security, and Performance menus.
2. When you are finished browsing through the menus, log out of Enterprise Manager Database Express.
3. Log out of the terminal window

**Practices for Lesson 7: Overview**

### Overview

In these practices, you will learn more about initialization parameters. You will also learn how to view diagnostic information.

**Practice 7-1: Investigating Initialization Parameter Files**

### Overview

In this practice, you investigate how the Oracle Database server uses initialization parameter files to start the database instance.

### Assumptions

You are logged in as the oracle user.

### Tasks

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

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

privilege.

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

…

SQL>

1. Locate the default SPFILE for your database instance by using the SHOW PARAMETER command. The results show that the SPFILE is in the ORACLE\_HOME/dbs directory. The output in the code box has been formatted for legibility.

--------------------- -----------

spfile string

VALUE

---------------------------------------------------------------

/u01/app/oracle/product/18.0.0/dbhome\_1/dbs/spfileORCL.ora

SQL>

TYPE

NAME

SQL> **SHOW PARAMETER spfile**

1. View the init.ora file. This is the sample text initialization parameter file (PFILE) provided with the Oracle Database installation.
   1. Use the SQL\*Plus HOST command to return to the operating system prompt.

SQL> **host**

[oracle@MYDBCS ~]$

* 1. Change to the ORACLE\_HOME/dbs directory and use the ls command to list the contents of the directory.

[oracle@MYDBCS ~]$ **cd $ORACLE\_HOME/dbs**

[oracle@MYDBCS dbs]$ **ls**

[oracle@MYDBCS dbs]$

|  |  |  |  |
| --- | --- | --- | --- |
| hc\_ORCL.dat | init.ora | lkORCL | snapcf\_ORCL.f |
| hc\_ORCL.dat | initORCL.ora | lkORCL | spfileORCL.ora |
| initORCL.ora | initORCL.ora.ORG | orapwORCL | tmp\_PDB1.xml |

Notice that the SPFILE (spfileORCL.ora) and init.ora files are stored here. The naming convention for an SPFILE is spfile<SID>.ora.

* 1. Use the cat or more command to view the contents of the sample text initialization parameter file (PFILE), init.ora.

[oracle@MYDBCS dbs]$ **more init.ora**

#

# $Header: rdbms/admin/init.ora /main/25 2015/05/14 15:02:30 kasingha Exp $

#

# Copyright (c) 1991, 2015, Oracle and/or its affiliates. All rights reserved.

# NAME

# init.ora # FUNCTION # NOTES

# MODIFIED

…

################################################################ ##############

# Example INIT.ORA file #

# This file is provided by Oracle Corporation as a starting point for

# customizing the Oracle Database installation for your site. #

# NOTE: The values that are used in this file are example values only.

# You may want to adjust those values for your specific requirements.

# You might also consider using the Database Configuration Assistant

# tool (DBCA) to create a server-side initialization parameter file

# and to size your initial set of tablespaces. See the # Oracle Database 2 Day DBA guide for more information.

################################################################ ###############

# Change '<ORACLE\_BASE>' to point to the oracle base (the one you specify at

# install time)

db\_name='ORCL'

… undo\_tablespace='UNDOTBS1'

# You may want to ensure that control files are created on separate physical

# devices

control\_files = (ora\_control1, ora\_control2) compatible ='11.2.0'

[oracle@MYDBCS dbs]$

* 1. Now use the cat or more command to view the text initialization parameter file,

initORCL.ora.

[oracle@MYDBCS dbs]$ **more initORCL.ora**

…

\*.audit\_file\_dest='/u01/app/oracle/admin/ORCL/adump'

\*.audit\_trail='db'

\*.compatible='18.0.0'

\*.control\_files='/u02/app/oracle/oradata/ORCL/control01.ctl','/u 03/app/oracle

/fast\_recovery\_area/ORCL/control02.ctl'

\*.db\_block\_checking='MEDIUM'

\*.db\_block\_checksum='TYPICAL'

\*.db\_block\_size=8192

\*.db\_domain='588436052.oraclecloud.internal'

…

\*.remote\_login\_passwordfile='EXCLUSIVE'

\*.sec\_protocol\_error\_trace\_action='LOG'

\*.service\_names='ORCL.588436052.oraclecloud.internal'

\*.sga\_target=2756471040

\*.sql92\_security=TRUE

\*.undo\_tablespace='UNDOTBS1'

[oracle@MYDBCS dbs]$

In Database Cloud Service, this text file is created automatically. You can create a text initialization parameter file from the SPFILE by using the following command:

CREATE PFILE = 'init<*SID*>.ora' FROM SPFILE;

* 1. Return to SQL\*Plus.

[oracle@MYDBCS dbs]$ **exit**

exit

SQL>

1. If the database server doesn't find an SPFILE, then the text initialization parameter file will be used. Now you’ll set up a test to see how the search works when you start the database instance.
   1. Shut down the database instance in IMMEDIATE mode.

SQL> **SHUTDOWN IMMEDIATE**

SQL> shutdown immediate Database closed.

Database dismounted.

ORACLE instance shut down.

SQL>

* 1. Use the HOST command to return to an operating system prompt.

SQL> **host**

[oracle@MYDBCS ~]$

* 1. Change to the ORACLE\_HOME/dbs directory. [oracle@MYDBCS ~]$ **cd $ORACLE\_HOME/dbs** [oracle@MYDBCS dbs]$
  2. Rename the spfileORCL.ora file to orig\_spfileORCL.ora. Renaming this file

will take it out of the search order for parameter files when you start up the database instance. Instead, the database server will automatically find the initORCL.ora file (PFILE) to start the database instance.

[oracle@MYDBCS dbs]$ **mv spfileORCL.ora orig\_spfileORCL.ora**

[oracle@MYDBCS dbs]$

* 1. Return to SQL\*Plus.

[oracle@MYDBCS dbs]$ **exit**

exit

SQL>

* 1. Start the database instance by using the STARTUP command.

SQL> **STARTUP**

ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes

|  |  |  |
| --- | --- | --- |
| Variable Size | 704643072 | bytes |
| Database Buffers | 1979711488 | bytes |
| Redo Buffers  Database mounted. | 74985472 | bytes |
| Database opened. |  |  |
| SQL> |  |  |

* 1. Verify that the database instance was started with your PFILE by issuing the SHOW PARAMETER spfile command. The value is null, which means the database instance was started with a PFILE.

------------------------------------ ----------- ---------------

spfile string

SQL>

VALUE

TYPE

SQL> **SHOW PARAMETER spfile**

NAME

1. Configure the database instance to once again start with the SPFILE.
   1. Shut down the database instance in IMMEDIATE mode.

SQL> **SHUTDOWN IMMEDIATE**

Database closed. Database dismounted.

ORACLE instance shut down.

SQL>

* 1. Use the HOST command to return to the operating system.

SQL> **host**

[oracle@MYDBCS ~]$

* 1. Change to the ORACLE\_HOME/dbs directory. [oracle@MYDBCS ~]$ **cd $ORACLE\_HOME/dbs** [oracle@MYDBCS dbs]$
  2. Rename the orig\_spfileORCL.ora file to spfileORCL.ora.

[oracle@MYDBCS dbs]$ **mv orig\_spfileORCL.ora spfileORCL.ora**

[oracle@MYDBCS dbs]$

* 1. Return to SQL\*Plus.

[oracle@MYDBCS dbs]$ **exit**

exit

SQL>

* 1. Start the database instance by using the STARTUP command.

SQL> **STARTUP**

ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes Variable Size 704643072 bytes

Database Buffers 1979711488 bytes Redo Buffers 74985472 bytes Database mounted.

Database opened.

SQL>

* 1. Verify that the database instance was started with the SPFILE.

------------------------------------ ----------- ---------------

spfile string

VALUE

----------------------------------------------------------------

/u01/app/oracle/product/18.0.0/dbhome\_1/dbs/spfileORCL.ora

SQL>

VALUE

TYPE

SQL> **SHOW PARAMETER spfile**

NAME

1. Exit SQL\*Plus.

SQL> **EXIT**

Disconnected from Oracle Database 18c Enterprise Edition Release

18.0.0.0.0 - Production Version 18.1.0.0.0

[oracle@MYDBCS ~]$

**Practice 7-2: Viewing Initialization Parameters by Using SQL\*Plus**

### Overview

In this practice, you view initialization parameters (parameters) by using SQL\*Plus. You do this in two ways:

* By using the SHOW PARAMETER command
* By querying the following views: V$PARAMETER, V$SPPARAMETER, V$PARAMETER2

and V$SYSTEM\_PARAMETER

### Assumptions

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

### Tasks

##### View Basic Parameters

In this section, you view basic parameters by using the SHOW PARAMETER command. Basic parameters are those parameters that you are likely to modify.

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

privilege.

$ **sqlplus / as sysdba**

..

SQL>

1. View the values of the DB\_NAME and DB\_DOMAIN parameters. Together, these values create the global database name.
   1. View the value of the DB\_NAME parameter. This parameter specifies the current database identifier of up to eight characters. If you have multiple databases, the value of this parameter should match the Oracle instance identifier of each one to avoid confusion with other databases running on the system.

ORCL

string

db\_name

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER db\_name**

NAME

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

------- ------------------------------

VALUE

TYPE

NAME

------------------------

SQL> **SHOW PARAMETER db\_domain**

db\_domain string 588436052.oraclecloud.internal SQL>

1. View the DB\_RECOVERY\_FILE\_DEST and DB\_RECOVERY\_FILE\_DEST\_SIZE parameters. These parameters set the location of the fast recovery area and its size.

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

The DB\_RECOVERY\_FILE\_DEST\_SIZE parameter specifies (in bytes) the hard limit on the total space to be used by target database recovery files created in the fast recovery area.

##### SQL> SHOW PARAMETER db\_recovery\_file\_dest

NAME TYPE VALUE

---------------------- ------ ----------------------------------

db\_recovery\_file\_dest string /u03/app/oracle/fast\_recovery\_area db\_recovery\_file\_dest\_size big integer 4G

SQL>

1. View the SGA\_TARGET and SGA\_MAX\_SIZE parameters.

SGA\_TARGET specifies the total amount of SGA memory available to a database instance and SGA\_MAX\_SIZE sets a maximum size for the SGA.

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

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

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

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

------------------------------------ ----------- ---------------

allow\_group\_access\_to\_sga boolean FALSE

lock\_sga boolean FALSE

pre\_page\_sga boolean TRUE

VALUE

TYPE

SQL> **SHOW PARAMETER sga**

NAME

1048576

integer

big integer 2640M big integer 0

big integer 2640M

sga\_max\_size sga\_min\_size sga\_target

unified\_audit\_sga\_queue\_size

SQL>

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

UNDOTBS1

string

undo\_tablespace

SQL>

------------------------------------ ----------- ---------------

VALUE

SQL> **SHOW PARAMETER undo\_tablespace**

NAME TYPE

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

18.0.0

FALSE

string

boolean

compatible noncdb\_compatible

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER compatible**

NAME

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

------------------------------------ ----------

control\_files string

VALUE

----------------------------------------------------------------

/u02/app/oracle/oradata/ORCL/control01.ctl,

/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl

SQL>

TYPE

SQL> **SHOW PARAMETER control\_files**

NAME

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

1

1

0

1

4000

4

300

integer integer integer integer integer integer

integer

aq\_tm\_processes db\_writer\_processes gcs\_server\_processes global\_txn\_processes job\_queue\_processes log\_archive\_max\_processes processes

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER processes**

NAME

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

0

0

0

0

472

integer integer integer integer integer

integer

java\_max\_sessionspace\_size java\_soft\_sessionspace\_limit license\_max\_sessions license\_sessions\_warning sessions shared\_server\_sessions

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER sessions**

NAME

* 1. View the TRANSACTIONS parameter. This parameter specifies how many rollback segments to bring online when the UNDO\_MANAGEMENT initialization parameter is equal to MANUAL. A transaction is assigned to a rollback segment when the transaction starts, and it can't change for the life of the transaction. A transaction table exists in the rollback segment header with limited space, limiting how many transactions a single

segment can support. Therefore, X number of concurrent transactions require at least X number of rollback segments. With Oracle Automatic Undo Management, the database creates rollback segments, brings them online, takes them offline, and drops them as needed.

519

5

integer

integer

transactions transactions\_per\_rollback\_segment

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER transactions**

NAME

1. View the configuration for the DB\_FILES initialization parameter. This parameter specifies the maximum number of database files that can be opened for this database. The range of values is OS-dependent.

500

integer

db\_files

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER db\_files**

NAME

##### View Advanced Parameters

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

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

------------------------------------ ----------- --------------

commit\_logging string

SQL>

VALUE

TYPE

SQL> **SHOW PARAMETER commit\_logging**

NAME

1. View the COMMIT\_WAIT parameter. This parameter is used to control when the redo for a commit is flushed to the redo logs. There is no default value.

------------------------------------ ----------- ---------------

commit\_wait string

SQL>

VALUE

TYPE

SQL> **SHOW PARAMETER commit\_wait**

NAME

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

------------------------------------ ----------- ---------------

shared\_pool\_size big integer 0

SQL>

VALUE

SQL> **SHOW PARAMETER shared\_pool\_size**

NAME TYPE

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

8192

integer

db\_block\_size

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER db\_block\_size**

NAME

1. View the DB\_CACHE\_SIZE initialization parameter. You configure this parameter to specify the size of the standard block buffer cache (default buffer pool). The range of values is at least 4 MB times the number of CPUs. Smaller values are automatically rounded up to this value. The default value is zero if the SGA\_TARGET initialization parameter is set, otherwise the larger of 48 MB or (4 MB\*CPU\_COUNT).

------------------------------------ ----------- ---------------

db\_cache\_size big integer 0

SQL>

VALUE

TYPE

SQL> **SHOW PARAMETER db\_cache\_size**

NAME

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

##### SQL> SHOW PARAMETER undo\_management

NAME TYPE VALUE

------------------------------------ ----------- ---------------

undo\_management string AUTO SQL>

1. View the MEMORY\_TARGET and MEMORY\_MAX\_TARGET parameters. MEMORY\_TARGET specifies the Oracle system-wide usable memory. The database server tunes memory to the MEMORY\_TARGET value, reducing or enlarging the SGA and PGA as needed. MEMORY\_MAX\_TARGET sets a maximum value for MEMORY\_TARGET.

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

* 1. View the MEMORY\_TARGET parameter.

------------------------------------ ----------- ---------------

memory\_target big integer 0

SQL>

VALUE

TYPE

SQL> **SHOW PARAMETER memory\_target**

NAME

* 1. View the MEMORY\_MAX\_TARGET parameter.

------------------------------------ ----------- ---------------

memory\_max\_target big integer 0

SQL>

VALUE

SQL> **SHOW PARAMETER memory\_max\_target**

NAME TYPE

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

------------------------------------ ----------- ---------------

pga\_aggregate\_target big integer 1837647360

SQL>

VALUE

SQL> **SHOW PARAMETER pga\_aggregate\_target**

NAME TYPE

##### Query Views for Parameter Values

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

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

SQL> **SET PAGES 100**

SQL> **SELECT table\_name FROM dict WHERE table\_name LIKE '%PARAMETER%';**

TABLE\_NAME

---------------------------------------------------------------- USER\_ADVISOR\_EXEC\_PARAMETERS

USER\_ADVISOR\_PARAMETERS

USER\_ADVISOR\_SQLW\_PARAMETERS USER\_XS\_ACL\_PARAMETERS ALL\_APPLY\_PARAMETERS ALL\_CAPTURE\_PARAMETERS ALL\_XS\_ACL\_PARAMETERS

… V$PARAMETER\_VALID\_VALUES V$SPPARAMETER V$SYSTEM\_PARAMETER V$SYSTEM\_PARAMETER2

V$SYSTEM\_RESET\_PARAMETER V$SYSTEM\_RESET\_PARAMETER2

66 rows selected.

SQL>

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

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

--------------------------------------- -------- ------------ NUM NUMBER

NAME VARCHAR2(80)

TYPE NUMBER

VALUE VARCHAR2(4000)

DISPLAY\_VALUE VARCHAR2(4000)

DEFAULT\_VALUE VARCHAR2(255)

ISDEFAULT VARCHAR2(9)

ISSES\_MODIFIABLE VARCHAR2(5)

ISSYS\_MODIFIABLE VARCHAR2(9)

Type

Null?

SQL> **DESCRIBE v$parameter**

Name

ISPDB\_MODIFIABLE VARCHAR2(5)

ISINSTANCE\_MODIFIABLE VARCHAR2(5)

ISMODIFIED VARCHAR2(10)

ISADJUSTED VARCHAR2(5)

ISDEPRECATED VARCHAR2(5)

ISBASIC VARCHAR2(5)

DESCRIPTION VARCHAR2(255)

UPDATE\_COMMENT VARCHAR2(255)

HASH NUMBER

CON\_ID NUMBER

SQL>

* 1. Query NAME, ISSYS\_MODIFIABLE, and VALUE in the V$PARAMETER view. The query returns many rows.

The TRANSACTIONS parameter is static as indicated by FALSE in the ISSYS\_MODIFIABLE column. The PLSQL\_WARNINGS parameter is dynamic as indicated by IMMEDIATE in the ISSYS\_MODIFIABLE column.

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

433 rows selected.

SQL>

strong

not allowed

IMMEDIATE

IMMEDIATE

300

FALSE

FALSE

lock\_name\_space processes

…

multishard\_query\_data\_consistency multishard\_query\_partial\_results

----------------------------------- ------------ ---------------

VALUE

ISSYS\_MOD

NAME

SQL> **SELECT name, issys\_modifiable, value FROM v$parameter;**

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

SQL> **COLUMN name FORMAT A30**

SQL> **COLUMN value FORMAT A10**

SQL> **SELECT name, value FROM v$parameter**

2 **WHERE name LIKE '%pool%';**

9 rows selected.

SQL>

0

0

0

0

0

26843545

0

shared\_pool\_size large\_pool\_size java\_pool\_size streams\_pool\_size shared\_pool\_reserved\_size memoptimize\_pool\_size buffer\_pool\_keep buffer\_pool\_recycle

olap\_page\_pool\_size

------------------------------ ----------

VALUE

NAME

1. Explore the V$SPPARAMETER view. This view contains information about the contents of the server parameter file. If a server parameter file was not used to start the instance, each row of the view will contain FALSE in the ISSPECIFIED column.
   1. View the columns in the V$SPPARAMETER view by using the DESCRIBE command.

----------------------------------------- -------- ------------ FAMILY VARCHAR2(80)

SID VARCHAR2(80)

NAME VARCHAR2(80)

TYPE VARCHAR2(11)

VALUE VARCHAR2(255)

DISPLAY\_VALUE VARCHAR2(255)

ISSPECIFIED VARCHAR2(6)

ORDINAL NUMBER

UPDATE\_COMMENT VARCHAR2(255)

CON\_ID NUMBER

SQL>

Type

Null?

SQL> **DESCRIBE v$spparameter**

Name

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

SQL> **SELECT name, value FROM v$spparameter;**

NAME VALUE

--------------------------------- ----------

lock\_name\_space

processes 300

sessions timed\_statistics timed\_os\_statistics

… shrd\_dupl\_table\_refresh\_rate

multishard\_query\_data\_consistency multishard\_query\_partial\_results

437 rows selected.

SQL>

1. Explore the V$PARAMETER2 view. This view contains information about the initialization parameters that are currently in effect for the session, with each parameter value appearing as a row in the view. A new session inherits parameter values from the instance-wide values displayed in the V$SYSTEM\_PARAMETER2 view.
   1. View the columns in the V$PARAMETER2 view by using the DESCRIBE command.

----------------------------------------- -------- ------------- NUM NUMBER

NAME VARCHAR2(80)

TYPE NUMBER

VALUE VARCHAR2(4000)

DISPLAY\_VALUE VARCHAR2(4000)

ISDEFAULT VARCHAR2(6)

ISSES\_MODIFIABLE VARCHAR2(5)

ISSYS\_MODIFIABLE VARCHAR2(9)

ISPDB\_MODIFIABLE VARCHAR2(5)

ISINSTANCE\_MODIFIABLE VARCHAR2(5)

ISMODIFIED VARCHAR2(10)

ISADJUSTED VARCHAR2(5)

ISDEPRECATED VARCHAR2(5)

ISBASIC VARCHAR2(5)

DESCRIPTION VARCHAR2(255)

ORDINAL NUMBER

UPDATE\_COMMENT VARCHAR2(255)

CON\_ID NUMBER

SQL>

Type

Null?

SQL> **DESCRIBE v$parameter2**

Name

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

438 rows selected.

SQL>

300

472

TRUE 0 TRUE

60

strong

not allowed

VALUE

----------

NAME

------------------------------

lock\_name\_space processes sessions timed\_statistics

timed\_os\_statistics resource\_limit

… shrd\_dupl\_table\_refresh\_rate

multishard\_query\_data\_consistency

multishard\_query\_partial\_results

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

1. Explore the V$SYSTEM\_PARAMETER view. This view contains information about the initialization parameters that are currently in effect for the instance.
   1. View the columns in the V$SYSTEM\_PARAMETER view by using the DESCRIBE

command.

----------------------------------------- -------- ------------- NUM NUMBER

NAME VARCHAR2(80)

TYPE NUMBER

VALUE VARCHAR2(4000)

DISPLAY\_VALUE VARCHAR2(4000)

DEFAULT\_VALUE VARCHAR2(255)

ISDEFAULT VARCHAR2(9)

ISSES\_MODIFIABLE VARCHAR2(5)

ISSYS\_MODIFIABLE VARCHAR2(9)

ISPDB\_MODIFIABLE VARCHAR2(5)

ISINSTANCE\_MODIFIABLE VARCHAR2(5)

ISMODIFIED VARCHAR2(8)

ISADJUSTED VARCHAR2(5)

Type

Null?

SQL> **DESCRIBE v$system\_parameter**

Name

SQL>

VARCHAR2(5) VARCHAR2(5) VARCHAR2(255) VARCHAR2(255) NUMBER

NUMBER

ISDEPRECATED ISBASIC DESCRIPTION UPDATE\_COMMENT HASH

CON\_ID

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

multishard\_query\_data\_consistency strong multishard\_query\_partial\_results not allowed

457 rows selected.

SQL>

16

300

472

TRUE 0

TRUE

processes sessions timed\_statistics

timed\_os\_statistics

resource\_limit

…

parallel\_servers\_target common\_user\_prefix

--------------------------------- ----------

lock\_name\_space

VALUE

NAME

SQL> **SELECT name, value FROM v$system\_parameter;**

1. Exit SQL\*Plus.

SQL> **EXIT**

Disconnected from Oracle Database 18c Enterprise Edition Release

18.0.0.0.0 - Production Version 18.1.0.0.0

[oracle@MYDBCS ~]$

**Practice 7-3: Modifying Initialization Parameters by Using SQL\*Plus**

### Overview

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

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

### Assumptions

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

### Tasks

##### Modify a Session-Level Parameter

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

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

$ **sqlplus / as sysdba**

.. SQL>

1. Learn about the NLS\_DATE\_FORMAT parameter by querying the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the NLS\_DATE\_FORMAT parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase.

SQL>

----------------- ----- --------- ------------- ---------------

nls\_date\_format TRUE FALSE TRUE

VALUE

ISSES ISSYS\_MOD ISPDB

NAME

SQL> **SELECT name, isses\_modifiable, issys\_modifiable,**

1. **ispdb\_modifiable, value**
2. **FROM v$parameter**
3. **WHERE name = 'nls\_date\_format';**
4. Find out the default date format for the database by querying the NLS\_TERRITORY parameter in the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the NLS\_TERRITORY parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase.

--------------------------- ------------------------------------

nls\_territory AMERICA

SQL>

VALUE

NAME

SQL> **SELECT name, value FROM v$parameter**

2 **WHERE name = 'nls\_territory';**

1. Connect to PDB1. Run a simple query against the sample data to view an example of the current default date format in use.
   1. Switch to PDB1 by using the ALTER SESSION command.

SQL> **ALTER SESSION SET container = PDB1;**

Session altered.

SQL>

* 1. Query the LAST\_NAME and HIRE\_DATE columns in the HR.EMPLOYEES table. Notice the date format is dd-mon-rr.

107 rows selected.

SQL>

07-JUN-02

07-JUN-02

07-JUN-02

07-JUN-02

17-JUN-03

21-SEP-05

13-JAN-01

03-JAN-06

21-MAY-07

King Kochhar De Haan Hunold Ernst

… Mavris Baer Higgins

Gietz

SQL> **SELECT last\_name, hire\_date FROM hr.employees;**

LAST\_NAME HIRE\_DATE

------------------------- ---------

1. Modify the NLS\_DATE\_FORMAT parameter to use the format mon dd yyyy by using the

ALTER SESSION command.

SQL> **ALTER SESSION SET nls\_date\_format = 'mon dd yyyy';**

Session altered.

SQL>

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL> **SELECT last\_name, hire\_date FROM hr.employees;**  LAST\_NAME HIRE\_DATE  ------------------------- ----------- | | | | |
| King |  | jun | 17 | 2003 |
| Kochhar |  | sep | 21 | 2005 |
| De Haan |  | jan | 13 | 2001 |
| Hunold |  | jan | 03 | 2006 |
| Ernst |  | may | 21 | 2007 |
| …  Mavris |  | jun | 07 | 2002 |
| Baer |  | jun | 07 | 2002 |
| Higgins |  | jun | 07 | 2002 |
| Gietz |  | jun | 07 | 2002 |
| 107 rows | selected. |  |  |  |
| SQL> |  |  |  |  |

1. Query the NLS\_DATE\_FORMAT parameter again by using the SHOW PARAMETER command. The value column now reflects the custom date format.

mon dd yyyy

string

nls\_date\_format

SQL>

------------------------------------ ----------- ---------------

VALUE

SQL> **SHOW PARAMETER nls\_date\_format**

NAME TYPE

1. Disconnect from PDB1 to end your session.

SQL> **DISCONNECT**

Disconnected from Oracle Database 18c Enterprise Edition Release

18.0.0.0.0 - Production Version 18.1.0.0.0

SQL>

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

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

Connected.

SQL>

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

107 rows selected.

SQL>

07-JUN-02

07-JUN-02

07-JUN-02

07-JUN-02

17-JUN-03

21-SEP-05

13-JAN-01

03-JAN-06

21-MAY-07

King Kochhar De Haan Hunold Ernst

… Mavris Baer Higgins

Gietz

SQL> **SELECT last\_name, hire\_date FROM hr.employees;**

LAST\_NAME HIRE\_DATE

------------------------- ---------

1. Query the NLS\_DATE\_FORMAT parameter again by using the SHOW PARAMETER command. The VALUE column no longer has the custom date format.

------------------------------------ ----------- ---------------

nls\_date\_format string

SQL>

VALUE

SQL> **SHOW PARAMETER nls\_date\_format**

NAME TYPE

##### Modify a Dynamic System-Level Parameter

In this section, you modify the JOB\_QUEUE\_PROCESSES parameter. This parameter specifies the maximum number of job slaves per database instance that can be created for the execution of DBMS\_JOB jobs and Oracle Scheduler (DBMS\_SCHEDULER) jobs.

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

SQL> **exit**

…

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

… SQL>

1. Learn about the JOB\_QUEUE\_PROCESSES parameter by querying the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the JOB\_QUEUE\_PROCESSES parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase.

SQL>

----------------------- ----- ------------------ ---------------

job\_queue\_processes FALSE IMMEDIATE 4000

VALUE

ISSES ISSYS\_MOD

NAME

SQL> **SELECT name, isses\_modifiable, issys\_modifiable, value**

**FROM v$parameter WHERE name = 'job\_queue\_processes';**

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

SQL> **ALTER SYSTEM SET job\_queue\_processes=15 SCOPE=BOTH;**

System altered.

SQL>

1. Use the SHOW PARAMETER command to verify that the JOB\_QUEUE\_PROCESSES parameter value is now equal to 15. Notice that only job was entered with the SHOW PARAMETER command instead of the full name, job\_queue\_processes. Remember, when you use the SHOW PARAMETER command, you don't have to enter the full name. The database server will find all parameters that contain the letters job. In this example, the database server found two parameters that contain the letters job: job\_queue\_processes and max\_datapump\_jobs\_per\_pdb. The query result indicates that the job\_queue\_processes value in memory is now 15.

15

100

integer

integer

job\_queue\_processes max\_datapump\_jobs\_per\_pdb

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER job**

NAME

1. Verify that the new value for the JOB\_QUEUE\_PROCESSES parameter persists after the database instance is restarted.
   1. Shut down the database instance with the IMMEDIATE mode.

SQL> **SHUTDOWN IMMEDIATE**

Database closed. Database dismounted.

ORACLE instance shut down.

SQL>

* 1. Start the database instance by using the STARTUP command.

SQL> **STARTUP**

ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes Variable Size 704643072 bytes

Database Buffers 1979711488 bytes Redo Buffers 74985472 bytes Database mounted.

Database opened.

SQL>

* 1. View the configuration for the JOB\_QUEUE\_PROCESSES parameter again by using the SHOW PARAMETER command. The value is 15, which proves that your change to the parameter persisted after the database instance was restarted.

15

100

integer

integer

job\_queue\_processes

max\_datapump\_jobs\_per\_pdb SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER job**

NAME

##### Modify a Static System-Level Parameter

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

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

SQL>

------------------------------- ----- ---------- ---------------

sec\_max\_failed\_login\_attempts FALSE FALSE 3

ISSES ISSYS\_MOD VALUE

NAME

SQL> **SELECT name, isses\_modifiable, issys\_modifiable, value**

**FROM v$parameter WHERE name = 'sec\_max\_failed\_login\_attempts';**

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

##### SQL> ALTER SYSTEM SET sec\_max\_failed\_login\_attempts = 2 COMMENT='Reduce for tighter security.' SCOPE=SPFILE;

System altered.

SQL>

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

3

integer

sec\_max\_failed\_login\_attempts

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER sec\_max**

NAME

1. Restart the database and then verify that the new value for the

SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter is updated.

* 1. Shut down the database instance with the IMMEDIATE mode.

SQL> **SHUTDOWN immediate**

Database closed. Database dismounted.

ORACLE instance shut down.

SQL>

* 1. Start the database instance by using the STARTUP command.

SQL> **STARTUP**

ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes Variable Size 704643072 bytes

Database Buffers 1979711488 bytes Redo Buffers 74985472 bytes Database mounted.

Database opened.

SQL>

* 1. View the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter value again by using the SHOW PARAMETER command. The query result indicates that the parameter's value was successfully changed to 2.

2

integer

sec\_max\_failed\_login\_attempts

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER sec\_max**

NAME

* 1. View the NAME and UPDATE\_COMMENT columns in the V$PARAMETER view for the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter. Notice that the comment you added is stored in this view. The results below are formatted for easier reading.

-------------------------------- -------------------------------

sec\_max\_failed\_login\_attempts Reduce for tighter security.

SQL>

UPDATE\_COMMENT

NAME

SQL> **SELECT name, update\_comment**

**FROM v$parameter WHERE name='sec\_max\_failed\_login\_attempts';**

1. Exit SQL\*Plus and close the terminal window.

SQL> **EXIT**

**Practice 7-4: Modifying Initialization Parameters by Using Enterprise Manager Database Express**

### Overview

In this practice, you modify the JOB\_QUEUE\_PROCESSES initialization parameter (parameter) by using Oracle Enterprise Manager Express (EM Express). In a previous practice, you changed this parameter value to 15. In this practice, you change its value back to 4000.

### Assumptions

### Tasks

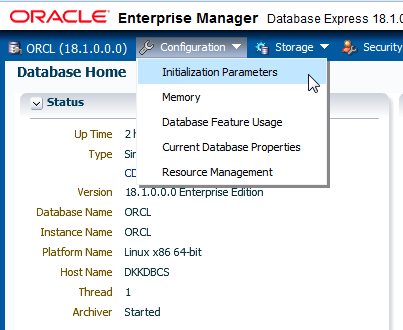
1. On your Linux desktop, create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
   1. Open a terminal window.
   2. Use ssh to create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

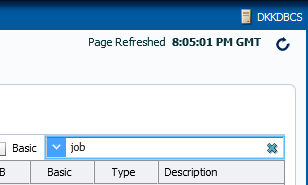
[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

[oracle@MYDBCS ~]$

1. On your Linux desktop, open Firefox. Launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. On the Enterprise Manager Database Express login page, enter the following:
   1. In the User Name field, enter **SYS**.
   2. In the Password field, enter the password you specified when you created the database deployment.
   3. Leave the Container Name field empty.
   4. Select **as sysdba**.
   5. Click Login.
3. On the Home page, expand the **Configuration** menu and then select **Initialization Parameters**.

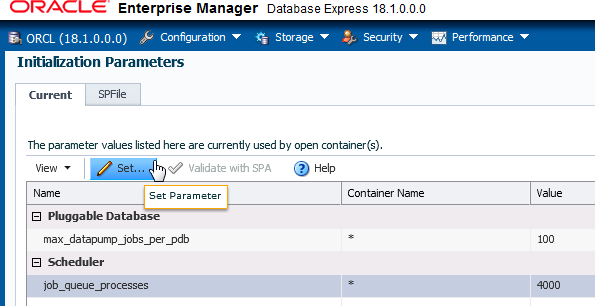


1. View the list of parameters by scrolling down.
2. In the filter box, enter **job** (in this interface, capitalization doesn't matter) to filter the parameters list to show only those parameters that contain the word job.

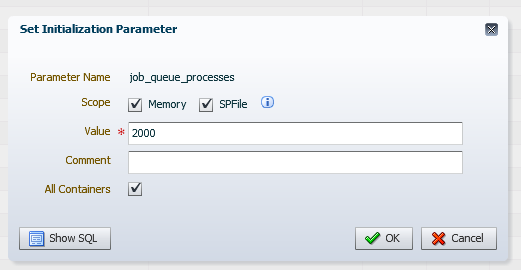


1. The JOB\_QUEUE\_PROCESSES initialization parameter should be listed. Select

**job\_queue\_processes** and click **Set**.



1. In the Set Initialization Parameter dialog box, enter **2000** in the **Value** field.



1. Click Show SQL, view the SQL statement that is going to be executed, review the questions and answers below, and then click **OK**.



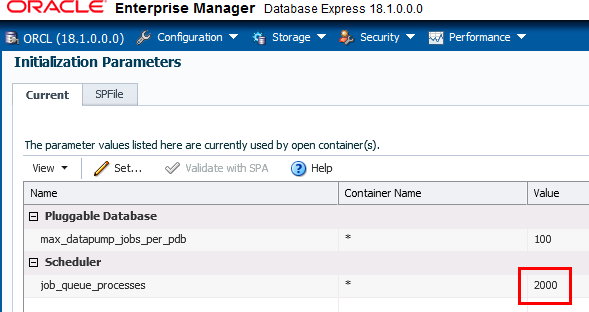
* 1. Question: What does the SCOPE=BOTH clause mean?

Answer: The parameter value update is persistent now in memory and across the instance shutdown as the value is written to the SPFILE.

* 1. Question: Is the change to the parameter's value applied to PDBs or only to the CDB?

Answer: The parameter's updates are applied for the instance, hence for all containers of the CDB instance. Nevertheless, some parameters may hold values specific for some PDBs.

1. In the Set Initialization Parameter dialog box, click **OK**.
2. In the Confirmation dialog box, verify that the message states "Parameter job\_queue\_processes successfully updated," and click **OK**.
3. Notice that the job\_queue\_processes value is updated to 2000.



1. In the upper-right corner, click **Log Out** to log out of EM Express.

**Practice 7-5: Shutting Down and Starting Up the Oracle Database**

### Overview

This practice lets you look more closely at shutting down and starting up your Oracle database instance.

### Assumptions Tasks

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

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

$ **sqlplus / as sysdba**

…

SQL>

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

SQL> **SHUTDOWN**

Database closed. Database dismounted.

ORACLE instance shut down.

SQL>

1. Show the current user. Note that SQL\*Plus is still running and the current user is SYS.

SQL> **SHOW USER**

USER is "SYS"

SQL>

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

SQL> **SHOW con\_name**

ERROR:

ORA-01034: ORACLE not available Process ID: 0

Session ID: 0 Serial number: 0

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

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

SQL> **STARTUP NOMOUNT**

ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes Variable Size 704643072 bytes

Database Buffers 1979711488 bytes Redo Buffers 74985472 bytes

SQL>

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

SQL> **ALTER DATABASE MOUNT;**

Database altered.

SQL>

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

SQL> **ALTER DATABASE OPEN;**

Database altered.

SQL>

1. Show the current container name.

SQL> **SHOW con\_name**

CON\_NAME

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

SQL>

1. Show the current user.

SQL> **SHOW user**

USER is "SYS" SQL>

1. Check whether PDB1 is open by querying the OPEN\_MODE column in the V$PDBS view.

------ ---------- ----------

1. PDB$SEED READ ONLY
2. PDB1 READ WRITE

SQL>

OPEN\_MODE

CON\_ID NAME

SQL> **COLUMN con\_id FORMAT 999**

SQL> **COLUMN name FORMAT A10**

SQL> **SELECT con\_id, name, open\_mode FROM v$pdbs;**

1. Did you expect PDB1 to be open? By default, PDBs are mounted when a CDB is opened. However, in Oracle Database Cloud Service a database event trigger opens all PDBs after startup. You can execute a query against DBA\_TRIGGERS to view information about the trigger that opens the PDBs.

STARTUP

AFTER EVENT

OPEN\_ALL\_PLUGGABLES

BEGIN

-------------------- -------------------- ---------------------- TRIGGER\_BODY

----------------------------------------------------------------

TRIGGERING\_EVENT

TRIGGER\_TYPE

TRIGGER\_NAME

SQL> **SELECT trigger\_name, trigger\_type, triggering\_event,**

1. **trigger\_body**
2. **FROM dba\_triggers**
3. **WHERE trigger\_name LIKE 'OPEN%';**

EXECUTE IMMEDIATE 'ALTER PLUGGABLE DATABASE ALL OPEN'; END;

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

**Practice 7-6: Viewing Diagnostic Information**

### Overview

In this practice, you perform the following tasks:

* + Examine the structure of the Automatic Diagnostic Repository (ADR)
  + View the alert log two ways—first through a text editor and then using the Automatic Diagnostic Repository Command Interpreter (ADRCI)
  + Enable DDL logging and log some DDL statements in the DDL log file

The alert log is a file that provides a chronological log of database messages and errors. It is automatically created and stored, by default, in the Automatic Diagnostic Repository (ADR) on the database server in the $ORACLE\_BASE/diag/rdbms/<db\_name>/<SID>/trace directory.

ADRCI is an Oracle command-line utility that enables you to investigate problems, view health check reports, and package and upload first-failure data to Oracle Support. You can also use the utility to view the names of the trace files in the ADR and to view the alert log. ADRCI has a rich command set that you can use interactively or in scripts.

The DDL log file contains one log record for each DDL statement.

### Assumptions

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

### Tasks

##### View the ADR Directories

The Automatic Diagnostics Repository (ADR) is a hierarchical file-based repository for handling diagnostic information. You can navigate the contents of ADR by using your operating system's command line, file browsing tools, or Oracle's ADR Command Interpreter (ADRCI). ADRCI is preferred for many tasks.

In this section, you locate the XML and text-only versions of the alert log by querying the

V$DIAG\_INFO view.

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

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

… SQL>

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

SQL> **SELECT name, value FROM v$diag\_info;**

-------------------- -------------------------------------------

Diag Enabled TRUE

ADR Base /u01/app/oracle

ADR Home /u01/app/oracle/diag/rdbms/orcl/ORCL

Diag Trace /u01/app/oracle/diag/rdbms/orcl/ORCL/trace Diag Alert /u01/app/oracle/diag/rdbms/orcl/ORCL/alert Diag Incident /u01/app/oracle/diag/rdbms/orcl/ORCL/incident Diag Cdump /u01/app/oracle/diag/rdbms/orcl/ORCL/cdump Health Monitor /u01/app/oracle/diag/rdbms/orcl/ORCL/hm Default Trace File

/u01/app/oracle/diag/rdbms/orcl/ORCL/trace/ORCL\_ora\_2600.trc Active Problem Count 1

Active Incident Count 6

11 rows selected.

SQL>

VALUE

NAME

1. Exit SQL\*Plus.

SQL> **EXIT**

##### Use an Editor to View the Alert Log

1. View the XML version of the alert log. The log.xml file is the XML version of the alert log.
   1. Browse to the /u01/app/oracle/diag/rdbms/orcl/ORCL/alert directory.

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

[oracle@MYDBCS alert]$

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

[oracle@MYDBCS alert]$ **ls**

log.xml

[oracle@MYDBCS alert]$

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

[oracle@MYDBCS alert]$ **more log.xml**

<msg time='2018-03-07T22:19:08.858+00:00' org\_id='oracle' comp\_id='rdbms'

msg\_id='opistr\_real:1244:2538814769' type='NOTIFICATION' group='startup'

level='16' host\_id='MYDBCS' host\_addr='10.18.24.38'

pid='8090' version='1' con\_uid='1'

con\_id='1' con\_name='CDB$ROOT'>

<txt>Starting ORACLE instance (normal) (OS id: 8090)

…

1. View the text-only version of the alert log.
   1. Change to the /u01/app/oracle/diag/rdbms/orcl/ORCL/trace directory.

[oracle@MYDBCS alert]$ **cd**

**/u01/app/oracle/diag/rdbms/orcl/ORCL/trace**

[oracle@MYDBCS trace]$

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

ORCL\_ora\_12783.trm

ORCL\_ora\_12785.trc

ORCL\_ora\_12783.trc

[oracle@MYDBCS trace]$ **ls**

alert\_ORCL.log ORCL\_ora\_27673.trc alert\_ORCL\_v20180310.log.gz

ORCL\_ora\_27673.trm

ORCL\_aqpc\_10857.trc ORCL\_ora\_27674.trc

…

[oracle@MYDBCS trace]$

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

[oracle@MYDBCS trace]$ **tail -500 alert\_ORCL.log**

2018-03-15T20:41:22.507272+00:00

db\_recovery\_file\_dest\_size of 6144 MB is 33.63% used. This is a user-specified limit on the amount of space that will be used by this

database for recovery-related files, and does not reflect the amount of

space available in the underlying filesystem or ASM diskgroup.

…

Pluggable database PDB1 opened read write 2018-03-16T15:10:48.837547+00:00

Completed: ALTER PLUGGABLE DATABASE ALL OPEN

Starting background process CJQ0 Completed: ALTER DATABASE OPEN 2018-03-16T15:10:49.521562+00:00

CJQ0 started with pid=65, OS id=4909

[oracle@MYDBCS trace]$

##### Use ADRCI to View the Alert Log

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

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

ADRCI: Release 18.0.0.0.0 - Production on Fri Mar 16 19:17:45 2018

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

ADR base = "/u01/app/oracle" adrci>

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

adrci> **SHOW ALERT**

ADR Home = /u01/app/oracle/diag/rdbms/orcl/ORCL:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

Output the results to file: /tmp/alert\_11237\_1404\_ORCL\_1.ado 2018-03-07 22:19:08.858000 +00:00

Starting ORACLE instance (normal) (OS id: 8090)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*

Dump of system resources acquired for SHARED GLOBAL AREA (SGA) Per process system memlock (soft) limit = 128G

Expected per process system memlock (soft) limit to lock SHARED GLOBAL AREA (SGA) into memory: 2642M

Available system pagesizes: 4K, 2048K

…

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

2018-03-16 15:10:45.273000 +00:00

Opening pdb with no Resource Manager plan active 2018-03-16 15:10:47.226000 +00:00

Pluggable database PDB1 opened read write

2018-03-16 15:10:48.837000 +00:00

Completed: ALTER PLUGGABLE DATABASE ALL OPEN

Starting background process CJQ0 Completed: ALTER DATABASE OPEN

CJQ0 started with pid=65, OS id=4909

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

Starting ORACLE instance (normal) (OS id: 8090)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*

Dump of system resources acquired for SHARED GLOBAL AREA (SGA) Per process system memlock (soft) limit = 128G

Expected per process system memlock (soft) limit to lock SHARED GLOBAL AREA (SGA) into memory: 2642M

Available system pagesizes:

4K, 2048K

Supported system pagesize(s):

PAGESIZE AVAILABLE\_PAGES EXPECTED\_PAGES ALLOCATED\_PAGES ERROR(s)

NONE NONE

4K Configured 4 675844

2048K 0 1321 0

RECOMMENDATION:

* 1. For optimal performance, configure system with expected number

of pages for every supported system pagesize prior to the next instance restart operation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\* LICENSE\_MAX\_SESSION = 0

LICENSE\_SESSIONS\_WARNING = 0

Initial number of CPU is 2

Number of processor cores in the system is 2 Number of processor sockets in the system is 1 search hit BOTTOM, continuing at TOP

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

2018-03-07 22:21:04.104000 +00:00

Using default pga\_aggregate\_limit of 3505 MB 2018-03-07 22:21:06.471000 +00:00

MOUNT

ALTER DATABASE

.... (PID:9128): Redo network throttle feature is disabled at mount time

Successful mount of redo thread 1, with mount id 2299076813 Database mounted in Exclusive Mode

Lost write protection disabled

.... (PID:9128): Using STANDBY\_ARCHIVE\_DEST parameter default value as USE\_DB\_RECOVERY\_FILE\_DEST [krsd.c:17695]

Completed: ALTER DATABASE MOUNT

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

ALTER DATABASE OPEN

Ping without log force is disabled: instance mounted in exclusive mode.

Buffer Cache Full DB Caching mode changing from FULL CACHING DISABLED to FULL CACHING ENABLED

Crash Recovery excluding pdb 2 which was cleanly closed. Crash Recovery excluding pdb 3 which was cleanly closed. 2018-03-07 22:21:08.617000 +00:00

Endian type of dictionary set to little LGWR (PID:9095): STARTING ARCH PROCESSES

Starting background process ARC0

1. Exit the vi editor by entering **:q** and pressing Enter.
2. Exit adrci and close the terminal window.

adrci > **exit**

[oracle@MYDBCS trace]$

##### Log DDL Statements in the DDL Log File

1. Determine if DDL logging is enabled in PDB1. If not, enable it by setting the value for the

ENABLE\_DDL\_LOGGING initialization parameter to TRUE.

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

[oracle@MYDBCS trace]$ **sqlplus / as sysdba**

…

SQL>

* 1. Switch to PDB1.

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

Session altered.

SQL>

* 1. Issue the SHOW PARAMETER command to view the value for ENABLE\_DDL\_LOGGING. In Oracle Database Cloud Service, ENABLE\_DDL\_LOGGING is set to TRUE by default. The default value for ENABLE\_DDL\_LOGGING is FALSE in non-Cloud installations.

TRUE

boolean

enable\_ddl\_logging

SQL>

------------------------------------ ----------- ---------------

VALUE

SQL> **SHOW PARAMETER enable\_ddl\_logging**

NAME TYPE

* 1. If DDL logging was not enabled, you could enable it for just this session by using the

ALTER SESSION command.

SQL> ALTER SESSION SET enable\_ddl\_logging = TRUE; Session altered.

SQL>

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

SQL> **CREATE TABLE TEST (name varchar2(15));**

Table created.

SQL> **DROP TABLE TEST;**

Table dropped.

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS trace]$

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

[oracle@MYDBCS trace]$ **cd**

**/u01/app/oracle/diag/rdbms/orcl/ORCL/log**

[oracle@MYDBCS log]$

1. List the contents of the log directory.

[oracle@MYDBCS log]$ **ls**

ddl ddl\_ORCL.log debug debug.log hcs imdb test

[oracle@MYDBCS log]$

1. View the ddl\_ORCL.log file by using the cat command. Your output will be different from the output shown below.

[oracle@MYDBCS log]$ **cat ddl\_ORCL.log**

2018-03-16T19:31:30.795903+00:00

diag\_adl:CREATE TABLE TEST (name varchar2(15) ) 2018-03-16T19:31:57.762139+00:00

diag\_adl:DROP TABLE TEST

[oracle@MYDBCS log]$

1. Change to the ddl directory and list the contents. The XML version of the DDL log file (log.xml) is located here.

[oracle@MYDBCS log]$ **cd ddl** [oracle@MYDBCS ddl]$ **ls** log.xml

[oracle@MYDBCS ddl]$

1. Close the terminal window.

**Practices for Lesson 8: Understanding Oracle Net Services**

## Practices for Lesson 8: Overview

### Overview

In these practices, you will explore the configuration of the default listener and create a static listener.

## Practice 8-1: Exploring the Default Listener

### Overview

In this practice, you explore the configuration for the default listener, LISTENER, and dynamic service registration.

### Assumptions

You are logged in as the oracle user.

### Tasks

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

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

$ **sqlplus / as sysdba**

…

SQL>

1. View the initialization parameters used during dynamic service registration.
   1. INSTANCE\_NAME: This parameter identifies the database instance name. It defaults to the Oracle System Identifier (SID) of the database instance. The results show that the database instance name is ORCL, which you named during installation.

ORCL

string

instance\_name

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER instance\_name**

NAME

* 1. SERVICE\_NAMES: This parameter identifies the service names that users can use in their connection strings to connect to the database instance. By default, the service name takes on the same name as the global database name, which is a combination of the DB\_NAME parameter and the DB\_DOMAIN parameter. The SERVICE\_NAMES parameter can accept multiple comma-separated values if you want to provide users with a variety of service names for the database instance. Doing so helps you control and monitor different user groups in Oracle Database Resource Manager.

|  |  |  |
| --- | --- | --- |
| SQL> **SHOW PARAMET**  NAME | **ER service\_names**  TYPE VALUE | |
| ---------------- | ------- | ------------------------------------- |
| service\_names | string | ORCL.588436052.oraclecloud.internal |
| SQL> |  |  |

* 1. LOCAL\_LISTENER: This parameter specifies the alias names for local listeners that resolve to addresses in the tnsnames.ora file (or other address repository as configured for your system). If there are multiple aliases, they must be separated by commas and all values enclosed by one set of double quotation marks.

**Note:** In Oracle Database Cloud Service, an alias for the listener is not defined by default in the tnsnames.ora file, so the VALUE column for the LOCAL\_LISTENER initialization parameter is null.

------------------------------------ ----------- ---------------

local\_listener string

SQL>

VALUE

TYPE

SQL> **SHOW PARAMETER local\_listener**

NAME

* 1. REMOTE\_LISTENER: This parameter specifies the alias names for remote listeners (listeners on different machines than the database instance). If there are multiple aliases, they must be separated by commas and all values enclosed by one set of double quotation marks. The value is null, indicating that you do not have any remote listeners.

------------------------------------ ----------- ---------------

remote\_listener string

SQL>

VALUE

SQL> **SHOW PARAMETER remote\_listener**

NAME TYPE

1. Exit SQL\*Plus.

SQL> **EXIT**

Disconnected from Oracle Database 18c Enterprise Edition Release

18.0.0.0.0 - Production Version 18.1.0.0.0 [oracle@MYDBCS ~]$

1. View the tnsnames.ora file.
   1. Change directories to $ORACLE\_HOME/network/admin.

[oracle@MYDBCS ~]$ **cd $ORACLE\_HOME/network/admin**

[oracle@MYDBCS admin]$

* 1. List the files in this directory. The tnsnames.ora file is listed.

sqlnet.ora

samples

[oracle@MYDBCS admin]$ **ls**

listener.ora

tnsnames.ora

shrept.lst

listener.orapre\_vncr\_config

[oracle@MYDBCS admin]$

* 1. View the tnsnames.ora file by using the cat command (case matters).

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

ORCL =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = ORCL.588436052.oraclecloud.internal)

)

)

PDB1 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = PDB1.588436052.oraclecloud.internal)

)

)

[oracle@MYDBCS admin]$

1. View the listener.ora file by using the cat command. This file contains the listeners created on the machine. So far, you have one listener, which is the default listener. When you start the Listener Control utility, it connects to the named listener or the default listener (LISTENER) if you leave out the name. To connect, the Listener Control utility obtains the protocol address(es) for the listener by resolving the listener name with one of the following mechanisms:

* listener.ora file in the directory specified by the TNS\_ADMIN environment variable. This is why it's important to set the environment variables to the appropriate home before using the Listener Control utility, which you did at the beginning of this practice.
* listener.ora file in the $ORACLE\_HOME/network/admin directory
* Naming method, for example, a tnsnames.ora file

If the listener name is LISTENER and it cannot be resolved, a protocol address of TCP/IP, port 1521 is assumed.

[oracle@MYDBCS admin]$ **cat listener.ora**

# listener.ora Network Configuration File:

/u01/app/oracle/product/18.0.0/dbhome\_1/network/admin/listener.o ra

# Generated by Oracle configuration tools.

LISTENER = (DESCRIPTION\_LIST =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))

)

)

VALID\_NODE\_CHECKING\_REGISTRATION\_LISTENER=ON SSL\_VERSION = 1.2

[oracle@MYDBCS admin]$

1. Start the Listener Control utility. Without specifying a listener name, the utility assumes you want to connect to the default listener, LISTENER.

[oracle@MYDBCS admin]$ **lsnrctl**

LSNRCTL for Linux: Version 18.0.0.0.0 - Production on 16-MAR- 2018 20:50:06

Copyright (c) 1991, 2017, Oracle. All rights reserved. Welcome to LSNRCTL, type "help" for information.

LSNRCTL>

1. View information about the default listener by using the Listener Control utility.
   1. View the operations that are available by using the help command.

LSNRCTL>

services

save\_config exit

status

reload quit

stop version spawn

show\*

start servacls trace

set\*

LSNRCTL> **help**

The following operations are available

An asterisk (\*) denotes a modifier or extended command:

* 1. View the name of the current listener by using the SHOW command and the CURRENT\_LISTENER parameter. You can set the CURRENT\_LISTENER parameter to facilitate managing a particular listener. With it set to a particular listener, you don't need to specify the listener's name after each command. The utility will automatically execute all commands against that listener. If you want to work on a different listener, you can either set the CURRENT\_LISTENER parameter to the other listener's name by using the SET CURRENT\_LISTENER command or you can include the other listener's name after each command. Currently, the default listener is set to LISTENER.

LSNRCTL> **SHOW current\_listener**

Current Listener is LISTENER LSNRCTL>

* 1. View the status of LISTENER by using the status command. This command displays basic information about the listener, including its alias name (LISTENER), its version, when it was last started (Start Date), how long it’s been running for (Uptime), whether tracing is turned on (Trace Level), whether OS authentication is enabled (Security), whether SNMP is on, the location of the listener parameter file and log file, listener end points, the wallet directory, and a list of registered services and whether they are ready.

**Note:** Wallets are certificates, keys, and trustpoints processed by SSL that allow for secure connections.

LSNRCTL> **status**

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1521))) STATUS of the LISTENER

------------------------

Alias LISTENER

Version TNSLSNR for Linux: Version 18.0.0.0.0

- Production

Start Date 07-MAR-2018 22:17:39

Uptime 8 days 22 hr. 33 min. 51 sec

Trace Level off

Security ON: Local OS Authentication

SNMP OFF

Listener Parameter File

/u01/app/oracle/product/18.0.0/dbhome\_1/network/admin/listener.o ra

Listener Log File

/u01/app/oracle/diag/tnslsnr/MYDBCS/listener/alert/log.xml Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1521))) (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=5500))(Security=(my\_wallet\_ directory=/u01/app/oracle/admin/ORCL/xdb\_wallet))(Presentation=H TTP)(Session=RAW))

Services Summary...

Service "66db11a937912ac9e0532618120ab4b9.588436052.oraclecloud.internal " has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "ORCL.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "ORCL.588436052.oraclecloud.internalXDB" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "PDB1.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

The command completed successfully LSNRCTL>

Some of your service names will be different from the ones shown below.

* 1. To view additional details about the registered services, issue the SERVICES command. If the status value for the database instance associated with the database service is UNKNOWN, the LREG process is not communicating with the listener, and therefore, there is no dynamic service registration going on. If the status is READY, then dynamic service registration is going on.

LSNRCTL> **services**

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1521))) Services Summary...

Service "66db11a937912ac9e0532618120ab4b9.588436052.oraclecloud.internal " has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Handler(s):

"DEDICATED" established:0 refused:0 state:ready LOCAL SERVER

Service "ORCL.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Handler(s):

"DEDICATED" established:0 refused:0 state:ready LOCAL SERVER

Service "ORCL.588436052.oraclecloud.internalXDB" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Handler(s):

"D000" established:0 refused:0 current:0 max:1022 state:ready

DISPATCHER <machine: MYDBCS, pid: 4284> (ADDRESS=(PROTOCOL=tcp)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=40723))

Service "PDB1.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Handler(s):

"DEDICATED" established:0 refused:0 state:ready LOCAL SERVER

The command completed successfully

LSNRCTL>

The Handler(s) section contains the information about the dispatcher or the dedicated server process. In this case, it tells you the listener creates a DEDICATED server process for each service. The established and refused values count the number of successful and unsuccessful connections to the database service, and the state value tells you whether the handler is available (ready) or not.

* 1. Show the log status. The status is ON, which means the listener activity is being logged. Note that the domain that this command returns might differ from example.com.

LSNRCTL> **SHOW log\_status**

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1521)))

LISTENER parameter "log\_status" set to ON

The command completed successfully LSNRCTL>

* 1. Show the location of the log file. Note that the domain that this command returns might differ from example.com.

LSNRCTL> **SHOW log\_file**

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1521))) LISTENER parameter "log\_file" set to

/u01/app/oracle/diag/tnslsnr/MYDBCS/listener/alert/log.xml

The command completed successfully

LSNRCTL>

1. Exit the Listener Control utility.

LSNRCTL> **exit**

[oracle@MYDBCS admin]$

## Practice 8-2: Creating a Static Listener for a PDB

### Overview

In this practice, you create a listener named LISTENER\_PDB1 that listens on the non-default port 1562 for the PDB1 service. Configure the listener to use static service registration.

### Assumptions

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

### Tasks

##### Add an Entry to the listener.ora File

1. Copy the listener.ora file to listener.sav.

[oracle@MYDBCS admin]$ **cp listener.ora listener.sav**

[oracle@MYDBCS admin]$

1. Open listener.ora in an editor.

[oracle@MYDBCS admin]$ **vi listener.ora**

1. Add the following entry above the first entry, but below the two comments at the top. Examine the SID\_LIST\_LISTENER\_PDB1 section. Include the PDB service name in the GLOBAL\_DBNAME parameter, the database instance name in the SID\_NAME parameter, and the Oracle home directory in the ORACLE\_HOME parameter. Enter this information manually or copy and paste only one line at a time.

LISTENER\_PDB1 = (DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1562))

) SID\_LIST\_LISTENER\_PDB1 =

(SID\_LIST =

(SID\_DESC =

(GLOBAL\_DBNAME = pdb1.588436052.oraclecloud.internal) (SID\_NAME = ORCL)

(ORACLE\_HOME = /u01/app/oracle/product/18.0.0/dbhome\_1)

)

)

1. Save the file and close the editor.

##### Start LISTENER\_PDB1

Using the Listener Control utility, start LISTENER\_PDB1.

1. In the terminal window, start the Listener Control utility.

[oracle@MYDBCS admin]$ **lsnrctl**

LSNRCTL for Linux: Version 18.0.0.0.0 - Production on 06-APR- 2018 12:10:02

Copyright (c) 1991, 2017, Oracle. All rights reserved. Welcome to LSNRCTL, type "help" for information.

LSNRCTL>

1. Check the status of LISTENER\_PDB1 by issuing the status command. Your results indicate "No listener" and "Connection refused" because you just created the listener and need to start it.

LSNRCTL> **status LISTENER\_PDB1**

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1562))) TNS-12541: TNS:no listener

TNS-12560: TNS:protocol adapter error TNS-00511: No listener

Linux Error: 111: Connection refused

LSNRCTL>

1. Start LISTENER\_PDB1 by issuing the start LISTENER\_PDB1 command. The results show that the listener has one service registered with it: pdb1.588436052.oraclecloud.internal. The status of the service is unknown. This is the case for static listeners. The LREG process doesn't update a static listener with database instance information.

LSNRCTL> **start LISTENER\_PDB1**

Starting /u01/app/oracle/product/18.0.0/dbhome\_1/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 18.0.0.0.0 - Production System parameter file is

/u01/app/oracle/product/18.0.0/dbhome\_1/network/admin/listener.o ra

Log messages written to

/u01/app/oracle/diag/tnslsnr/MYDBCS/listener\_pdb1/alert/log.xml

Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1562)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1562)))

STATUS of the LISTENER

------------------------

Alias listener\_pdb1

Version TNSLSNR for Linux: Version 18.0.0.0.0

- Production

Start Date 06-APR-2018 12:37:23

Uptime 0 days 0 hr. 0 min. 0 sec

Trace Level off

Security ON: Local OS Authentication

SNMP OFF

Listener Parameter File

/u01/app/oracle/product/18.0.0/dbhome\_1/network/admin/listener.o ra

Listener Log File

/u01/app/oracle/diag/tnslsnr/MYDBCS/listener\_pdb1/alert/log.xml Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1562))) Services Summary...

Service "pdb1.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status UNKNOWN, has 1 handler(s) for this service...

The command completed successfully LSNRCTL>

1. Exit the Listener Control utility.

LSNRCTL> **EXIT**

[oracle@MYDBCS admin]$

##### Connect to a Database Service through LISTENER\_PDB1

Test the connection to the PDB1.example.com service through the LISTENER\_PDB1 listener.

1. Using Easy Connect syntax, start SQL\*Plus and connect as the SYSTEM user to PDB1

using LISTENER\_PDB1. Make sure to specify the non-default port number 1562.

[oracle@MYDBCS admin]$ **sqlplus system/*password*@localhost:1562/pdb1.588436052.oraclecloud.intern al**

… SQL>

1. Show the current container name. The result is PDB1. You have successfully configured a static listener to support the PDB1 service.

SQL> **SHOW con\_name**

CON\_NAME

------------------------------ PDB1

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS admin]$

## Practice 8-3: Verifying the Net Service Name for PDB1

### Overview

In this practice, you verify that a net service name of **PDB1** is defined in the tnsnames.ora file. In Oracle Database Cloud Service (DBCS), the net service name for the PDB created by default is included in the tnsnames.ora. In a non-DBCS environment, you need to add the net service name entry to the tnsnames.ora file manually.

### Assumptions

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

### Tasks

1. View the entries in the tnsnames.ora file.
   1. If necessary, change the directory to $ORACLE\_HOME/network/admin.

[oracle@MYDBCS ~]$ **cd $ORACLE\_HOME/network/admin**

[oracle@MYDBCS admin]$

* 1. View the tnsnames.ora file by using the cat command. When your CDB and PDB were created, DBCA automatically created a net service name called ORCL.

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

ORCL =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = ORCL.588436052.oraclecloud.internal)

)

)

PDB1 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = PDB1.588436052.oraclecloud.internal)

)

)

[oracle@MYDBCS admin]$

Examine the ORCL entry. The connect descriptor encompasses all the information below ORCL, starting with (DESCRIPTION = ... to (SERVICE\_NAME = ORCL.588436052.oraclecloud.internal). The ADDRESS section describes the listener. The CONNECT\_DATA section describes the database service.

1. In Oracle Database Cloud Service, a net service name for the PDB is also defined. Test the Oracle Net service alias for PDB1 by using the tnsping utility. The last line in the results indicates that the connection is OK, which tells you that there is connectivity between the client and the Oracle Net Listener. It does not tell you whether the requested service is available.

[oracle@MYDBCS admin]$ **tnsping PDB1**

TNS Ping Utility for Linux: Version 18.0.0.0.0 - Production on 19-MAR-2018 16:15:00

Copyright (c) 1997, 2017, Oracle. All rights reserved. Used parameter files:

/u01/app/oracle/product/18.0.0/dbhome\_1/network/admin/sqlnet.ora

Used TNSNAMES adapter to resolve the alias

Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-588436052.oraclecloud.internal)(PORT

= 1521)) (CONNECT\_DATA = (SERVER = DEDICATED) (SERVICE\_NAME =

PDB1.588436052.oraclecloud.internal))) OK (0 msec)

[oracle@MYDBCS admin]$

1. Connect to PDB1 and verify the current container.
   1. Start SQL\*Plus and connect to PDB1 as the SYSTEM user by using the PDB1 net service name.

[oracle@MYDBCS admin]$ **sqlplus system/DBAdmin\_1@PDB1**

…

SQL>

Recall that in an earlier practice you used the Easy Connect syntax to make a direct connection to a PDB. Using a net service name such as PDB1 is much simpler.

* 1. Verify that the current container name is PDB1.

SQL> **SHOW con\_name**

CON\_NAME

------------------------------ PDB1

SQL>

1. Exit SQL\*Plus and disconnect from the compute node

SQL> **exit**

…

[oracle@MYDBCS admin]$ **exit**

## Practices for Lesson 9: Overview

### Overview

In these practices, you will create users and roles. You will grant privileges to users and roles.

## Practice 9-1: Creating Common and Local Users

### Overview

In this practice, you log on to the database in SQL\*Plus as the SYS user and create two types of administrators:

* CDB administrator named c##CDB\_ADMIN1: Create this user as a common user so that it exists in every container in the CDB. Grant this user the most powerful administrator privilege, the SYSDBA privilege, in all containers. This privilege enables c##CDB\_ADMIN1 to access containers whether they are open or not. Because most database operations don't require the SYSDBA privilege, also grant this user the DBA role and CREATE SESSION privilege in all containers so that the user can operate as a regular user too.
* PDB1 administrator named PDB1\_ADMIN1: Create this user as a local user in PDB1 and grant this user the DBA role and CREATE SESSION privilege. This grant will provide the necessary system and object privileges. All tasks required by this user must be performed on an open PDB.

##### Tip

It's good practice to create a user separate from SYS and SYSTEM to perform database administration tasks. Each DBA in your organization should have his or her own privileged account to aid in auditing. Keep in mind that when you connect with the SYSDBA privilege, the database shows you logged in as the SYS user, regardless of your actual username. Audit trails, however, will show your real username.

Organizations that need to implement the tightest security possible separate the database duties and create many accounts for each database administrator (DBA) distinctly named and use the security principle of least privileges. Only the minimum privileges needed to perform a job are given. If an administrator doesn't need access to data but still performs maintenance operations, you can grant that user the SYSOPER privilege instead. Also consider other administrative privileges, such as SYSDG, SYSKM, SYSBACKUP, and SYSRAC, as necessary.

### Assumptions

You are logged in as the oracle user.

### Tasks

##### Create c##CDB\_ADMIN1

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

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

privilege. This method of connecting uses OS authentication.

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

… SQL>

1. Create a common user named c##CDB\_ADMIN1 by using the CREATE USER command. Set the USERS tablespace as the default and TEMP as the temporary tablespace. Also, unlock the account so that c##CDB\_ADMIN1 can log in right away.

Important! To create a common user, you must start the user name with c## or C##, and you must include the CONTAINER=ALL clause so that the user's identity and password are created in all the containers.

Refer to *Course Practice Environment: Security Credentials* for the password value.

SQL> **CREATE USER c##CDB\_ADMIN1 IDENTIFIED BY *password***

##### CONTAINER=ALL DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp ACCOUNT UNLOCK;

User created.

SQL>

1. Grant c##CDB\_ADMIN1 the DBA role, the CREATE SESSION privilege, and the SYSDBA

privilege in all containers. This is an example of granting privileges and a role commonly.

SQL> **GRANT create session, dba, sysdba TO c##CDB\_ADMIN1 CONTAINER=ALL;**

Grant succeeded.

SQL>

1. Question: Can you use the following statement to complete the same operation (granting privileges and a role commonly)?

GRANT create session, dba TO c##CDB\_ADMIN1;

Answer: No, because without the CONTAINER=ALL clause, the CREATE SESSION privilege and DBA role are granted locally (in the root container only) to c##CDB\_ADMIN1, and not to each c##CDB\_ADMIN1 user in each PDB.

1. List the common users by querying the DBA\_USERS view. Scroll down and verify that

c##CDB\_ADMIN1 is included.

**SQL> SELECT DISTINCT username FROM dba\_users WHERE common='YES' ORDER BY username;**

USERNAME

---------------------------------------------------------------- ANONYMOUS

APPQOSSYS AUDSYS C##CDB\_ADMIN1 C##DBAAS\_BACKUP CTXSYS DBSFWUSER

… SYSTEM WMSYS XDB XS$NULL

39 rows selected.

SQL>

##### Compare Exercising and Not Exercising the SYSDBA Privilege

This section compares logging in as the c##CDB\_ADMIN1 user with and without the SYSDBA

privilege.

1. Disconnect from the root container.

SQL> **DISCONNECT**

… SQL>

1. Show the current user by issuing the SHOW USER command. You are not connected as any user.

SQL> **SHOW USER**

USER is "" SQL>

1. Connect to the root container as c##CDB\_ADMIN1 and exercise the SYSDBA privilege.

SQL> **CONNECT c##CDB\_ADMIN1/*password* AS SYSDBA**

Connected.

SQL>

1. Show the current container name.

SQL> **SHOW CON\_NAME**

CON\_NAME

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

SQL>

1. Show the current user. The current user is SYS, which means the c##CDB\_ADMIN1 user can now do anything that the SYS user can do.

Note: Audit trails will show the c##CDB\_ADMIN1 user, not SYS.

SQL> **SHOW USER**

USER is "SYS"

SQL>

1. View the list of privileges for the c##CDB\_ADMIN1 user by querying the SESSION\_PRIVS

static data dictionary view. Scroll down to view the privileges listed.

SQL> **SELECT \* FROM session\_privs ORDER BY privilege;**

PRIVILEGE

---------------------------------------- ADMINISTER ANY SQL TUNING SET ADMINISTER DATABASE TRIGGER

ADMINISTER KEY MANAGEMENT ADMINISTER RESOURCE MANAGER

…

UPDATE ANY TABLE USE ANY JOB RESOURCE

USE ANY SQL TRANSLATION PROFILE WRITE ANY ANALYTIC VIEW CACHE

253 rows selected.

SQL>

1. Disconnect from the root container.

SQL> **DISCONNECT**

…

SQL>

1. Connect to the root container as c##CDB\_ADMIN1 again, but this time, do not exercise the

SYSDBA privilege.

SQL> **CONNECT c##CDB\_ADMIN1/*password***

Connected.

SQL>

1. Show the current user. You are connected as c##CDB\_ADMIN1. Because you included the CONTAINER=ALL clause when granting the CREATE SESSION privilege and DBA role, c##CDB\_ADMIN1 can connect as a regular user to any open PDB and perform system and object operations that the DBA role allows.

SQL> **SHOW USER**

USER is "C##CDB\_ADMIN1" SQL>

1. View the list of privileges for the c##CDB\_ADMIN1 user by querying the SESSION\_PRIVS static data dictionary view. Scroll through the list of privileges. Notice that there are fewer privileges listed than when c##CDB\_ADMIN1 was connected with the SYSDBA privilege.

SQL> **SELECT \* FROM session\_privs ORDER BY privilege;**

PRIVILEGE

---------------------------------------- ADMINISTER ANY SQL TUNING SET ADMINISTER DATABASE TRIGGER

ADMINISTER RESOURCE MANAGER ADMINISTER SQL MANAGEMENT OBJECT

…

UPDATE ANY TABLE USE ANY JOB RESOURCE

USE ANY SQL TRANSLATION PROFILE

237 rows selected.

SQL>

1. Switch to PDB1 by issuing the ALTER SESSION command.

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

Session altered.

SQL>

1. Show the current container. It is PDB1.

SQL> **SHOW CON\_NAME**

CON\_NAME

------------------------------ PDB1

SQL>

##### Create the PDB1\_ADMIN User

You just connected to PDB1 as c##CDB\_ADMIN1. You need to be logged into PDB1 to create a local administrator for PDB1. The c##CDB\_ADMIN1 user can create the PDB1\_ADMIN1 user.

1. Create a local user named PDB1\_ADMIN1 by using the CREATE USER command. Set the USERS tablespace as the default and TEMP as the temporary tablespace. Also, unlock the account so that PDB1\_ADMIN1 can log in right away. Because this is a local user and not a common user, do not include the CONTAINER=ALL clause. Refer to *Course Practice Environment: Security Credentials* for the password value.

SQL> **CREATE USER PDB1\_ADMIN1 IDENTIFIED BY *password* DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp ACCOUNT UNLOCK;**

User created. SQL>

1. Grant PDB1\_ADMIN1 the DBA role and the CREATE SESSION privilege in PDB1 only. This is an example of granting a privilege and role locally.

SQL> **GRANT create session, dba TO PDB1\_ADMIN1;**

Grant succeeded.

SQL>

1. List the local user accounts for PDB1 by querying the DBA\_USERS view. The PDB1\_ADMIN1

account is included in the list.

SQL> **SELECT DISTINCT username FROM dba\_users WHERE common='NO' ORDER BY username;**

USERNAME

---------------------------------------------------------------- APEX\_050100

APEX\_INSTANCE\_ADMIN\_USER APEX\_LISTENER APEX\_PUBLIC\_USER APEX\_REST\_PUBLIC\_USER FLOWS\_FILES

HR PDB1\_ADMIN1 PDBADMIN SCOTT

10 rows selected.

SQL>

1. Disconnect c##CDB\_ADMIN1 from PDB1. SQL> **DISCONNECT**

…

SQL>

1. Connect to PDB1 as PDB1\_ADMIN1.

SQL> **CONNECT PDB1\_ADMIN1/*password*@PDB1**

Connected.

SQL>

1. Show the current user. You are connected as PDB1\_ADMIN1.

SQL> **SHOW USER**

USER is "PDB1\_ADMIN1" SQL>

1. View the list of privileges for PDB1\_ADMIN1 by querying the SESSION\_PRIVS view. The results below are only some of the privileges returned from the query.

SQL> **SELECT \* FROM session\_privs ORDER BY privilege;**

PRIVILEGE

---------------------------------------- ADMINISTER ANY SQL TUNING SET ADMINISTER DATABASE TRIGGER

ADMINISTER RESOURCE MANAGER

…

UPDATE ANY TABLE USE ANY JOB RESOURCE

USE ANY SQL TRANSLATION PROFILE

237 rows selected.

SQL>

1. Try to connect to the root container as the PDB1\_ADMIN1 user. This user does not have access to the root container, and therefore, you get an error message stating that the user has insufficient privileges. The c##CDB\_ADMIN1 user has the DBA role and CREATE SESSION privileges in all containers, including the root container. PDB1\_ADMIN has the same role and privilege, but only in PDB1.

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

ERROR:

ORA-01031: insufficient privileges

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS ~]$

## Practice 9-2: Creating a Local User for an Application

### Overview

In this practice, you log in to PDB1 as the local administrator (PDB1\_ADMIN1) and create a local user account called INVENTORY, which will own the new Inventory software application.

INVENTORY is an example of a user account that does not represent a person.

### Assumptions

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

### Tasks

##### Create the INVENTORY User Account

1. Start SQL\*Plus and connect to PDB1 as the PDB1\_ADMIN1 user.

[oracle@MYDBCS ~]$ **sqlplus PDB1\_ADMIN1/*password*@PDB1**

… SQL>

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

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

User created. SQL>

1. Grant the CREATE SESSION privilege to INVENTORY.

SQL> **GRANT CREATE SESSION TO INVENTORY;**

Grant succeeded.

SQL>

1. List the local user accounts for PDB1 by querying the DBA\_USERS view. The INVENTORY

account is included in the list.

SQL> **SELECT DISTINCT username FROM dba\_users WHERE common='NO' ORDER BY username;**

USERNAME

---------------------------------------------------------------- APEX\_050100

APEX\_INSTANCE\_ADMIN\_USER

APEX\_LISTENER APEX\_PUBLIC\_USER APEX\_REST\_PUBLIC\_USER FLOWS\_FILES

HR INVENTORY PDB1\_ADMIN1 PDBADMIN SCOTT

11 rows selected.

SQL>

##### Connect as INVENTORY and Verify Privileges

1. Disconnect PDB1\_ADMIN1 from PDB1.

SQL> **DISCONNECT**

… SQL>

1. Verify that the INVENTORY user account can connect to PDB1.

SQL> **CONNECT INVENTORY/*password*@PDB1**

Connected.

SQL>

1. List the privileges for INVENTORY by querying the SESSION\_PRIVS view. The results show that INVENTORY has the CREATE SESSION privilege.

SQL> **SELECT \* FROM session\_privs ORDER BY privilege;**

PRIVILEGE

---------------------------------------- CREATE SESSION

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

## Practice 9-3: Granting a Local Role (DBA) to PDBADMIN

### Overview

In this practice, you examine the default privileges and roles granted to the PDBADMIN user. PDBADMIN was created when the CDB and PDB1 were created. This user is intended to operate as the local PDB administrator.

After exploring, you grant PDBADMIN more power with the DBA role so that in later practices

PDBADMIN is able to create profiles, roles, and users.

### Assumptions

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

### Tasks

##### Explore the Privileges and Roles Granted to PDBADMIN

1. Start SQL\*Plus and connect as the SYS user with the SYSDBA privilege. Note: PDBADMIN does not have the required privileges to view data from the

DBA\_SYS\_PRIVS view in PDB1, which you will do in the next step.

$ **sqlplus / AS SYSDBA**

…

SQL>

1. List the system privileges granted to the PDBADMIN user by querying the DBA\_SYS\_PRIVS view. This view describes system privileges granted to users and roles. The results show that no system privileges are explicitly granted to PDBADMIN. However, there may be privileges granted through roles.

SQL> **SELECT \* FROM dba\_sys\_privs WHERE grantee='PDBADMIN';**

no rows selected

SQL>

1. List the roles granted to the PDBADMIN user by querying the CDB\_ROLE\_PRIVS view. This view describes the roles granted to all users and roles in the database. The results show that PDBADMIN is granted the PDB\_DBA role. Also, the ADMIN OPTION is enabled (ADM=YES), which means that PDBADMIN can grant the PDB\_DBA role to other users.

SQL> **col granted\_role format a10**

SQL> **SELECT granted\_role, admin\_option FROM cdb\_role\_privs WHERE grantee='PDBADMIN';**

GRANTED\_RO ADM

---------- ---

PDB\_DBA YES

SQL>

NO

DBA

1. List the system privileges granted to the PDB\_DBA role by querying the ROLE\_SYS\_PRIVS

view.

* 1. Switch to PDB1. You must be connected to PDB1 to retrieve data, and you must be connected as the SYS user.

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

Session altered.

SQL>

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

SQL> **SELECT privilege FROM role\_sys\_privs WHERE role='PDB\_DBA' ORDER BY privilege;**

PRIVILEGE

---------------------------------------- CREATE PLUGGABLE DATABASE

CREATE SESSION SET CONTAINER

SQL>

1. List the roles that are granted to the PDB\_DBA role by querying the DBA\_ROLE\_PRIVS view. The results show that the PDB\_DBA role is granted the CONNECT role.

SQL> **SELECT granted\_role FROM dba\_role\_privs WHERE grantee = 'PDB\_DBA';**

GRANTED\_RO

---------- CONNECT

SQL>

1. List the privileges granted to the CONNECT role by querying the ROLE\_SYS\_PRIVS view. The results show that the CONNECT role consists of the SET CONTAINER and CREATE SESSION privileges.

SQL> **SELECT privilege FROM role\_sys\_privs WHERE role='CONNECT' ORDER BY privilege;**

PRIVILEGE

---------------------------------------- CREATE SESSION

SET CONTAINER

SQL>

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

##### Grant the DBA Role to PDBADMIN

1. Grant the DBA role locally to PDBADMIN.

SQL> **GRANT dba TO pdbadmin;**

Grant succeeded.

SQL>

1. List the roles that are granted to PDBADMIN by querying the DBA\_ROLE\_PRIVS view. The results show that PDBADMIN is now granted the DBA and PDB\_DBA roles.

SQL> **SELECT granted\_role FROM dba\_role\_privs WHERE grantee = 'PDBADMIN' ORDER BY granted\_role;**

GRANTED\_RO

---------- DBA PDB\_DBA

SQL>

1. Exit SQL\*Plus and close the terminal window.

SQL> **EXIT**

…

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

## Practice 9-4: Using EM Express to Create a Local Profile

### Overview

In this practice, the PDBADMIN user (local administrator for PDB1) creates a local profile called HRPROFILE in to limit the amount of idle time users can have in the PDB. If a user is idle or forgets to log out after 60 minutes, the user session is ended.

In addition, the profile will be configured to automatically lock a database user account if it did not log on after a specified number of days. This locking mechanism is implemented through the INACTIVE\_ACCOUNT\_TIME user resource profile limit.

##### Tip

A local profile is a profile that resides in a single PDB. Therefore, to create one, you must log in to the PDB.

To log in to EM Express and perform administrative operations, such as creating profiles, PDBADMIN requires the EM\_EXPRESS\_ALL role. In the previous practice, you assigned the DBA role to PDBADMIN, which contains the EM\_EXPRESS\_ALL role.

### Assumptions

You are currently logged in as the oracle user.

You completed Practice 9-3 Granting the DBA Role to PDBADMIN.

### Tasks

##### Log in to Enterprise Manager Database Express (PDB1)

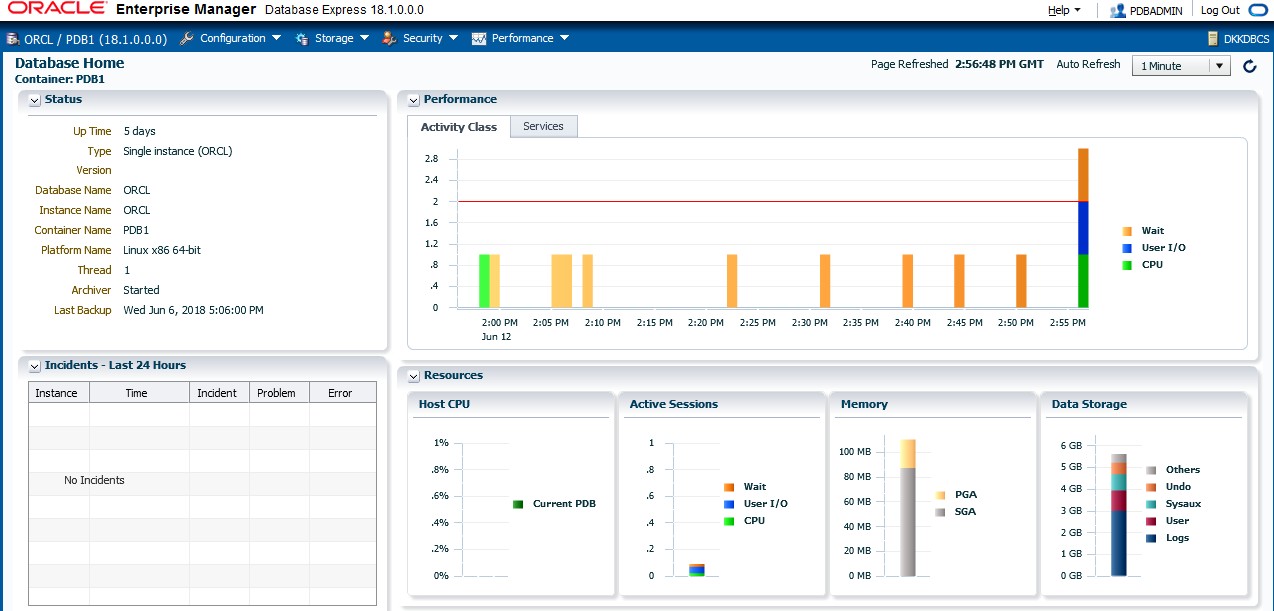
1. Create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
   1. Open a terminal window.
   2. Use ssh to create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

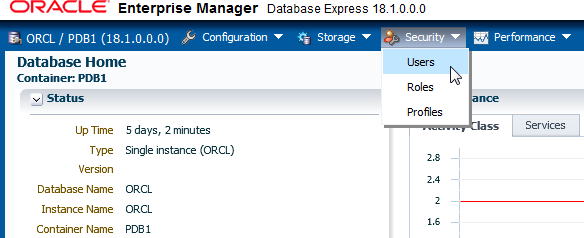
[oracle@MYDBCS ~]$

1. Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. On the Enterprise Manager Database Express login page, enter the following:
   1. In the User Name field, enter **PDBADMIN**.
   2. In the Password field, enter the password you specified when you created the database deployment.
   3. In the Container Name field, enter **PDB1**.
3. Click **Login**.
4. View the Database Home page, which shows information for PDB1.



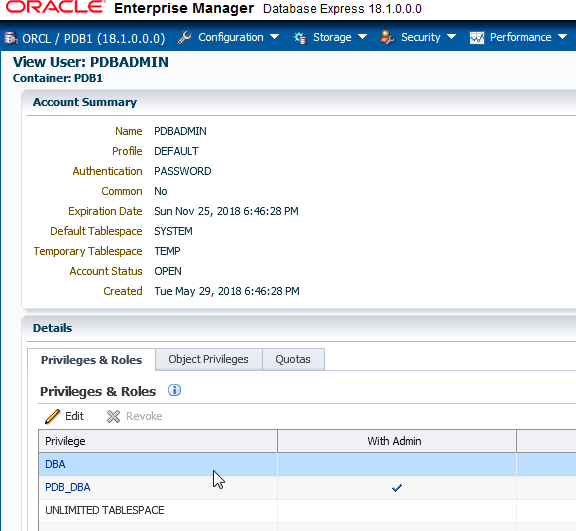
##### View Privileges and Roles for PDBADMIN

1. On the **Security** menu, select **Users**.



1. Scroll down the list of users and click **PDBADMIN**.
2. The View User: PDBADMIN page is displayed. On the **Privileges and Roles** tab, click

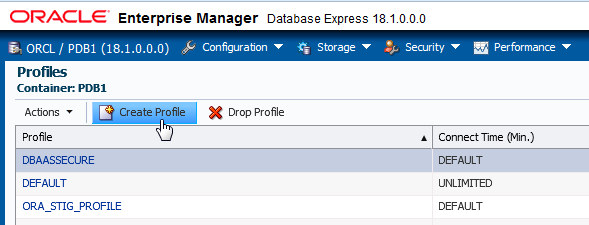
**DBA**.



1. Scroll down through the list of privileges and roles. Notice that any item in the list that is a role has a check mark in the Is Role column.
2. Use the search box to look for specific privileges and roles, for example, **CREATE PROFILE**.

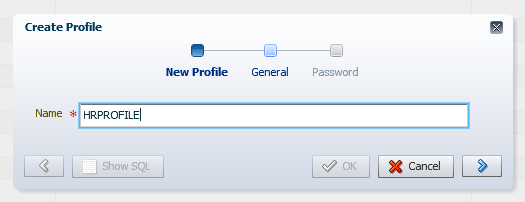
##### Create a Local Profile

1. On the **Security** menu, select **Profiles**.
2. Click **Create Profile**.

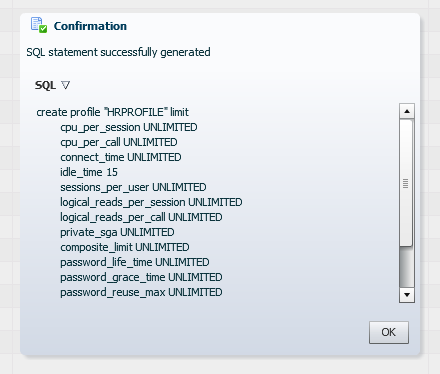


1. The Create Profile wizard is displayed. On the New Profile page, in the Name box, enter

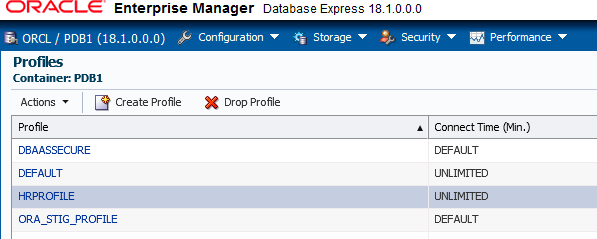
**HRPROFILE** and click Next (blue arrow).



1. On the General page, in the Idle Time (Minutes) drop-down list, select **15**. Leave all other fields set to the default value of Unlimited. Click **Next**. You will test the Idle Time setting in Practice 9-6 Creating Local Users, when you create users and assign them this profile.
2. On the Password page, review the password options. All should be set to default values of Unlimited or Null.
3. Click **Show SQL** to review the SQL command for this task. Click **OK** to close the window.

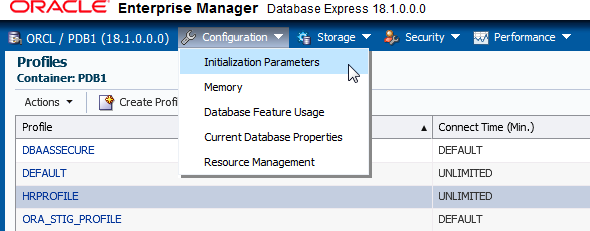


1. You are returned to the Password page. Click **OK** to close it.
2. On the Confirmation page, note the message "SQL statement has been processed successfully" and click **OK**.
3. Verify that HRPROFILE is in the list of profiles.



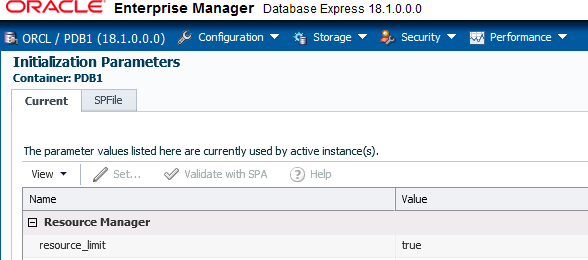
##### Set the RESOURCE\_LIMIT Initialization Parameter

1. On the **Configuration** menu, select **Initialization Parameters**.



1. In the **Name** search field, enter **resource\_limit**.
2. The RESOURCE\_LIMIT initialization parameter is listed in the table. Verify that the

RESOURCE LIMIT value is set to **true**.



1. If RESOURCE\_LIMIT is not set to true, perform the following steps:
   1. Click **Cancel** to return to the table.
   2. Select **resource\_limit** and click **Set**.
   3. In the Set Initialization Parameter dialog box, select the **true** option and click **OK**.
   4. In the Confirmation dialog box, click **OK**.
2. Click **Logout**, and close the browser window.

##### Modify the Profile so that Database User Accounts Will be Locked if Not Used in 10 Days

To lock database user accounts, modify the HRPROFILE profile to add the INACTIVE\_ACCOUNT\_TIME user resource profile limit. In this section, you use SQL\*Plus and learn by trial and error.

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Connect to PDB1 as the local DBA, PDBADMIN. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus PDBADMIN/*password*@PDB1**

… SQL>

1. Issue the ALTER PROFILE command to set the INACTIVE\_ACCOUNT\_TIME limit in the profile to 10 days.

SQL> **ALTER PROFILE hrprofile LIMIT INACTIVE\_ACCOUNT\_TIME 10;**

ALTER PROFILE hrprofile LIMIT INACTIVE\_ACCOUNT\_TIME 10

\*

ERROR at line 1:

ORA-02377: invalid profile limit INACTIVE\_ACCOUNT\_TIME

SQL>

1. Question: Is INACTIVE\_ACCOUNT\_TIME a valid profile limit? To find out, query the

DBA\_PROFILES view and confirm that INACTIVE\_ACCOUNT\_TIME is listed in the table.

-------- -------------------------------- -------------------- KERNEL COMPOSITE\_LIMIT UNLIMITED

KERNEL SESSIONS\_PER\_USER UNLIMITED

LIMIT

RESOURCE RESOURCE\_NAME

SQL> **COL limit FORMAT A20**

SQL> **SELECT resource\_type, resource\_name, limit FROM dba\_profiles WHERE profile='HRPROFILE';**

PASSWORD PASSWORD\_LOCK\_TIME UNLIMITED

PASSWORD PASSWORD\_GRACE\_TIME UNLIMITED

PASSWORD INACTIVE\_ACCOUNT\_TIME DEFAULT

17 rows selected.

SQL>

UNLIMITED UNLIMITED UNLIMITED UNLIMITED

15

CPU\_PER\_SESSION CPU\_PER\_CALL LOGICAL\_READS\_PER\_SESSION LOGICAL\_READS\_PER\_CALL

IDLE\_TIME

KERNEL KERNEL KERNEL KERNEL KERNEL

…

Answer: The results show a resource named INACTIVE\_ACCOUNT\_TIME, so INACTIVE\_ACCOUNT\_TIME is a valid profile limit. Therefore, the error must have something to do with the value that you are trying to set for the profile limit.

1. Investigate by displaying the full error message that you received in step 3. To do this, issue the oerr command for the error number ora 2377. Notice that the error states the limit cannot be less than 15 days.

SQL>

as a permissible value.

limit cannot be less than 15 days.

\* For the PASSWORD\_GRACE\_TIME profile parameter,

parameter, the specified

//

//

0 is allowed

//

// \* For the INACTIVE\_ACCOUNT\_TIME profile

some additional restrictions apply:

//

profile parameters,

\*Action: Specify a limit greater than 0. For password

A value of 0 or lower was specified for the limit.

\*Cause:

//

//

SQL> **! oerr ora 2377**

02377, 00000, "invalid profile limit %s"

1. Set an appropriate limit. Because 10 is too low, use the lowest valid number instead (which is 15).

SQL> **ALTER PROFILE hrprofile LIMIT INACTIVE\_ACCOUNT\_TIME 15;**

Profile altered.

SQL>

1. Query the DBA\_PROFILES view again to confirm that the limit is set.

PASSWORD PASSWORD\_LOCK\_TIME UNLIMITED

PASSWORD PASSWORD\_GRACE\_TIME UNLIMITED

PASSWORD INACTIVE\_ACCOUNT\_TIME 15

17 rows selected.

SQL>

UNLIMITED

UNLIMITED UNLIMITED

COMPOSITE\_LIMIT

SESSIONS\_PER\_USER CPU\_PER\_SESSION

KERNEL KERNEL KERNEL

…

-------- -------------------------------- --------------------

LIMIT

RESOURCE RESOURCE\_NAME

SQL> **SELECT resource\_type, resource\_name, limit FROM dba\_profiles WHERE profile = 'HRPROFILE';**

1. Exit SQL\*Plus and close the terminal window.

SQL> **exit**

…

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

1. Question: What will a DBA have to do if a database user account gets locked due to this new limit?

Answer: The DBA will have to unlock the database user account to make it available for use again by issuing the following command:

ALTER USER acct\_user IDENTIFIED BY <password> ACCOUNT UNLOCK;

## Practice 9-5: Using EM Express to Create Local Roles

### Overview

In this practice, the PDBAMIN user uses Enterprise Manager Database Express (EM Express) to create the following local roles in PDB1:

* + HRCLERK: Grant this role the SELECT and UPDATE object privileges on the EMPLOYEES

table in the HR schema.

* + HRMANAGER: Grant this role the SELECT, UPDATE, INSERT, and DELETE object privileges on the entire HR schema.

You will assign these roles to local users in Practice 9-6 Creating Local Users.

### Assumptions

You are currently logged in as the oracle user.

You completed Practice 9-3 Granting the DBA Role to PDBADMIN.

### Tasks

##### Log in to Enterprise Manager Database Express (PDB1)

1. Create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
   1. Open a terminal window.
   2. Use ssh to create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

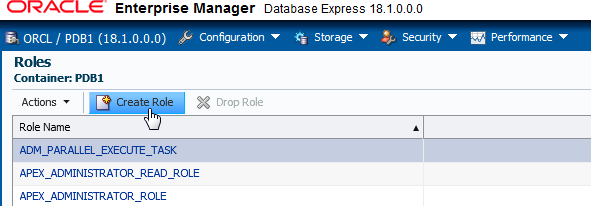
[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

[oracle@MYDBCS ~]$

1. Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. On the Enterprise Manager Database Express login page, enter the following:
   1. In the User Name field, enter **PDBADMIN**.
   2. In the Password field, enter the password you specified when you created the database deployment.
   3. In the Container Name field, enter **PDB1**.
3. Click Login.

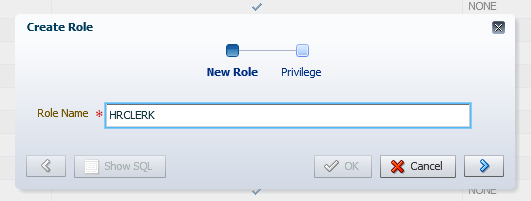
##### Create the HRCLERK Role

1. Select **Security** and then **Roles**.
2. The roles are listed in a table. Click **Create Role**.

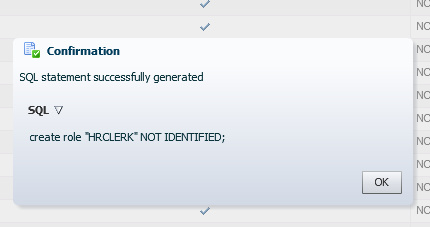


1. The Create Role Wizard is displayed. On the New Role page, in the Role Name box, enter

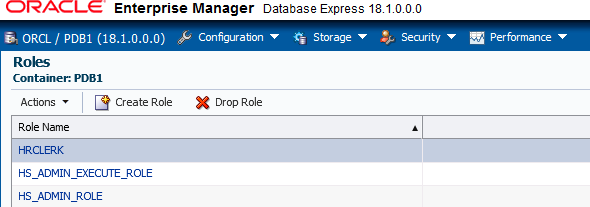
**HRCLERK** and click Next.



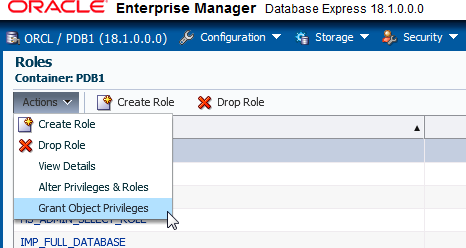
1. The Privilege page enables you to assign system privileges to the role. For this role, you will assign object privileges after the role is created. Click **Show SQL** to view the SQL statement.
2. After reviewing the SQL statement, click **OK** to return to the Privilege page.



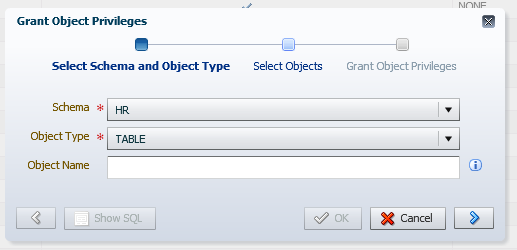
1. On the Privilege page, click **OK**.
2. In the Confirmation dialog box, click **OK**.
3. Verify that the **HRCLERK** role is listed in the table.
4. In the table, select the **HRCLERK** role.



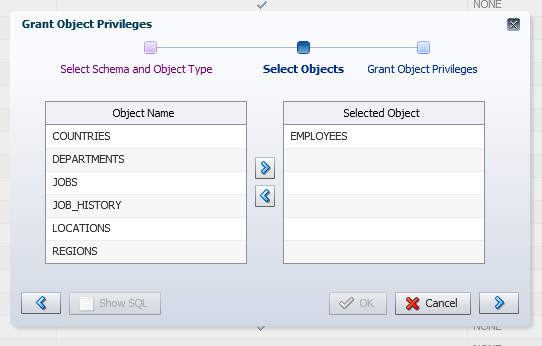
1. Select **Actions** and then **Grant Object Privileges**.



1. The Grant Object Privileges Wizard is displayed. On the Select Schema and Object Type page, do the following and click the arrow (Next).
   1. In the Schema drop-down list, select **HR**.
   2. In the Object Type drop-down list, ensure that **TABLE** is selected.
   3. Leave the Object Name box blank.

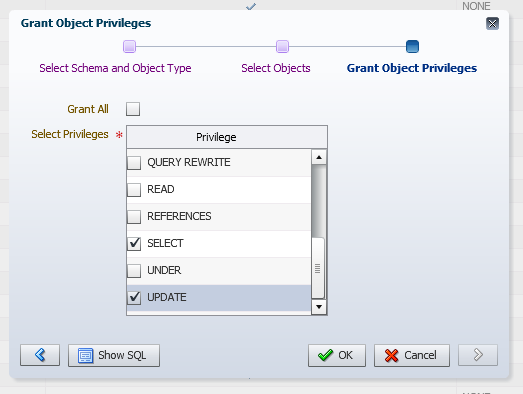


1. On the Select Objects page, select **EMPLOYEES** in the Object Name column and click the right arrow button to move it to the Selected Object column. Click the arrow (next).



1. On the Grant Object Privileges page, select the check boxes for the **SELECT** and **UPDATE**

privileges.



1. Click **Show SQL**.
2. The Confirmation dialog box displays the SQL commands that are generated to perform the grant. Click **OK**.



1. On the Grant Object Privileges page, click OK to complete the grant.
2. In the Confirmation dialog box, click **OK**. You are finished creating the **HRCLERK** role.

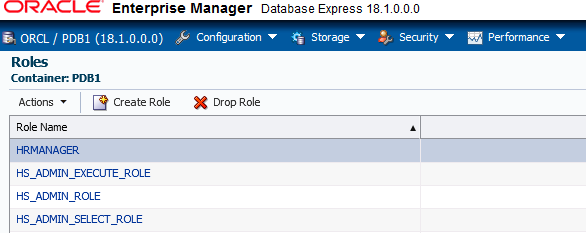
##### Create the HRMANAGER Role

The steps in this section are similar to the ones in the previous section.

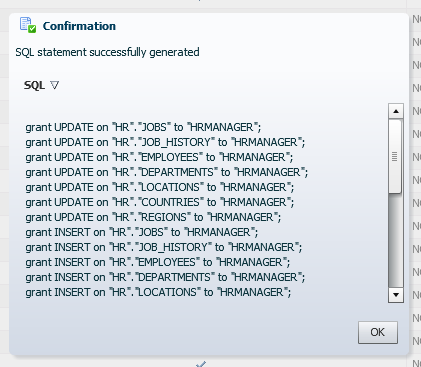
1. If you are not on the Roles page, select **Security** and then **Roles**. The roles are listed in a table.
2. Click **Create Role**.
3. The Create Role Wizard is displayed. On the New Role page, in the Role Name box, enter

**HRMANAGER** and click the arrow (next).

1. On the Privilege page, click **OK**.
2. In the Confirmation dialog box, click **OK**.
3. Verify that the **HRMANAGER** role is listed in the table.



1. In the table, select the HRMANAGER role. Select **Actions** and then **Grant Object Privileges**.
2. The Grant Object Privileges Wizard is displayed. On the Select Schema and Object Type page, do the following and then click the arrow (next).
   1. In the Schema drop-down list, select **HR**.
   2. In the Object Type drop-down list, select **TABLE**.
   3. Leave the Object Name box blank.
3. On the Select Objects page, select all the tables on the left (click and then Shift+click) and click the right arrow button to move them to the Selected Object list on the right. The following tables should be selected: REGIONS, LOCATIONS, JOB\_HISTORY, JOBS, EMPLOYEES, DEPARTMENTS, and COUNTRIES. Click the arrow (next).
4. On the Grant Object Privileges page, select the check boxes for the **DELETE**, **INSERT**, **SELECT**, and **UPDATE** privileges.
5. Click **Show SQL**.



1. The Confirmation dialog box displays the SQL commands that are generated to perform the grant. Click **OK**.
2. On the Grant Object Privileges page, click OK to complete the grant.
3. In the Confirmation dialog box, click OK. You have created the HRMANAGER role and granted the required privileges.
4. Click **Log Out**, and close the browser window.

## Practice 9-6: Using EM Express to Create Local Users

### Overview

In this practice, you log in to Enterprise Manager Database Express as the PDBADMIN user and create local user accounts in PDB1 according to the following table. Assign the profile named HRPROFILE to the accounts as well as various privileges and roles that you've already created in previous practices. Afterward, test the accounts by logging in to SQL\*Plus as each user. Also, test the idle time setting in HRPROFILE.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **User Account** | **Description** | **Privileges/Roles** | **Method to Use to Create User** |
| Jenny Goodman | JGOODMAN | A new HR | CREATE | EM Express |
|  |  | manager | SESSION |  |
|  |  |  | privilege |  |
|  |  |  | HRCLERK role |  |
|  |  |  | HRMANAGER role |  |
| David Hamby | DHAMBY | A new HR clerk | CREATE SESSION  privilege  HRCLERK role | EM Express |
| Rachel Pandya | RPANDYA | A new HR clerk | CREATE SESSION  privilege  HRCLERK role | SQL script with substitution variables |

### Assumptions

You are currently logged in as the oracle user.

You created a local profile in PDB1 named HRPROFILE and two local roles (HRCLERK and HRMANAGER) in PDB1. You also assigned the DBA role to the PDBADMIN user, which is the local administrator for PDB1. If you haven't done so, complete the following practices before starting this one:

* Practice 9-3 Granting the DBA Role to PDBADMIN
* Practice 9-4 Creating a Local Profile in EM Express and Locking Accounts
* Practice 9-5 Creating Local Roles in EM Express

### Tasks

##### Log In to EM Express as PDBADMIN

1. On your Linux desktop, create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
   1. Open a terminal window.
   2. Use ssh to create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

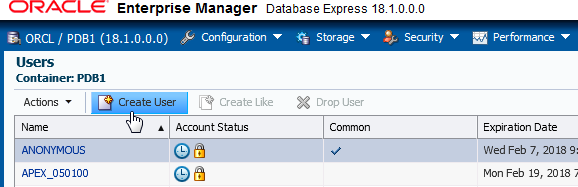
[oracle@MYDBCS ~]$

1. Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. On the Enterprise Manager Database Express login page, enter the following:
   1. In the User Name field, enter **PDBADMIN**.
   2. In the Password field, enter the password you specified when you created the database deployment.
   3. In the Container Name field, enter **PDB1**.
3. Click Login.

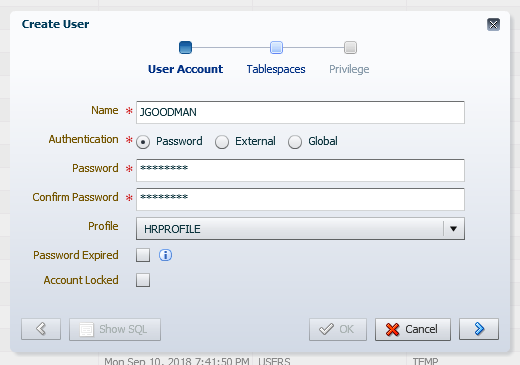
##### Create a User Account for Jenny Goodman

In this section, you create a user account named JGOODMAN by using the EM Express interface.

1. Select **Security** and then **Users**.
2. Click **Create User**.

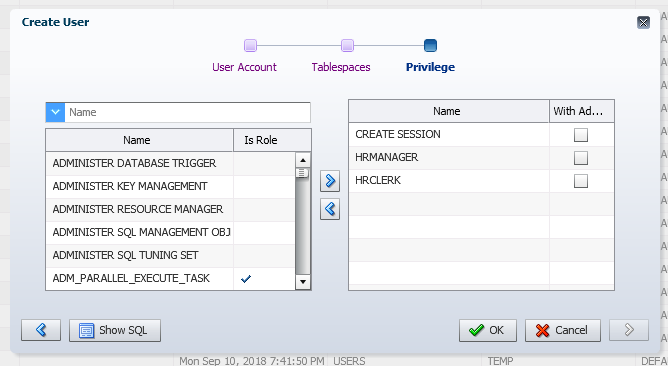


1. The Create User Wizard is displayed. On the User Account page, enter or select the following values and then click Next.
   * Name: Enter **JGOODMAN**
   * Authentication: Password
   * Password and Confirm Password: Refer to *Course Practice Environment: Security Credentials* for the password value.
   * Profile: Select **HRPROFILE**.
   * Leave the check boxes for Password Expired and Account Locked deselected.



1. On the Tablespaces page, ensure that the following tablespaces are selected and then click Next.
   * Default Tablespace: **USERS**
   * Temporary Tablespace: **TEMP**
2. On the Privilege page, select the **CREATE SESSION** privilege, **HRCLERK** role, and

**HRMANAGER** role and click the right arrow button to assign them to the user.



1. Click **Show SQL**.
2. After reviewing the SQL statement, click **OK**.

Note: In Practice 9-7 Configuring a Default Role for a User, you will assign the HRCLERK

role to be this user account's default role.

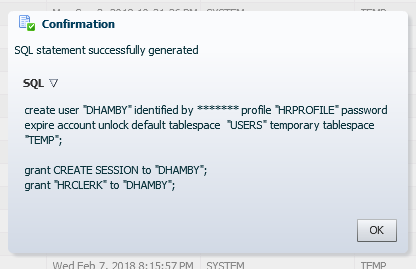
1. Click **OK** to execute the SQL statement.
2. In the Confirmation dialog box, click **OK**.
3. Verify that JGOODMAN is listed in the Users table. From here, you can see that JGOODMAN is a local user account (the Common column does not have a check mark), the account is unlocked and the password has not expired (there is no lock or clock in the Account Status column), and the account is assigned the HRPROFILE profile.

##### Create a User Account for David Hamby

In this section, you create another user account named DHAMBY by using the EM Express interface. While creating this user, you copy the SQL code to a text file so that in the next section, you can create more users by running a script.

1. If you are not on the Users page, expand **Security** and then select **Users**.
2. Click **Create User**.
3. The Create User Wizard is displayed. On the User Account page, enter or select the following values and then click Next.
   * Name: Enter **DHAMBY**
   * Authentication: **Password**
   * Password and Confirm Password: Refer to *Course Practice Environment: Security Credentials* for the password value.
   * Profile: Select **HRPROFILE**
   * Select the Password Expired check box to force the user to change his password at logon.
4. On the Tablespaces page, ensure that the following tablespaces are selected and click Next.
   * Default Tablespace: **USERS**
   * Temporary Tablespace: **TEMP**
5. On the Privilege page, select the **CREATE SESSION** privilege and **HRCLERK** role and click the right arrow button to assign them to the user.
6. Click **Show SQL**.
7. A Confirmation dialog box is displayed with the SQL for the CREATE USER and GRANT

statements. Do not close this dialog box.



1. Create a SQL script that contains the SQL statements displayed in the previous step. Turn the username and role values into substitution variables, rather than hard-coding them. You'll use this script to create future users.

Tip: You can create substitution variables in SQL scripts by using single ampersands (&) and/or double ampersands (&&). A single ampersand indicates to SQL\*Plus to prompt you to enter a value each time the substitution variable occurs in the script. A double ampersand indicates to SQL\*Plus to prompt you to enter a value only once for a substitution variable and use that same value for all occurrences of the variable in the script.

1. Copy to the clipboard the SQL statements in the previous step.
2. Minimize, but don't close, EM Express.
3. On the Linux desktop, select **Applications**, then **Accessories**, and then **gedit Text Editor**.
4. Select **Edit** and then **Paste** to paste the SQL statements.
5. Change every occurrence of DHAMBY to **&&username**. There are three occurrences.
6. Change the one occurrence of HRCLERK to **&&role**.
7. Replace the asterisks with the actual password. Refer to *Course Practice Environment: Security Credentials* for the password value.
8. Verify that the code looks like the following code. Don't worry if your GRANT statements are in the opposite order.

create user &&username identified by password profile "HRPROFILE" password

expire account unlock

default tablespace "USERS" temporary tablespace "TEMP";

grant CREATE SESSION to &&username; grant &&role to &&username;

1. Select **File** and then **Save As**.
2. Browse to **/home/oracle/labs**. In the Name box, enter **CreateHRUser.sql**. Click

**Save**. The file is saved and formatted.

1. Select **File** and then **Quit**.
2. Return to EM Express, click **OK** to close the Confirmation dialog box, and click **OK** to finish creating the user.
3. In the Confirmation dialog box, click **OK**.
4. Verify that **DHAMBY** is listed in the Users table. From here, you can see that DHAMBY is a local user account (the Common column does not have a check mark), the account is unlocked and the password has expired (there is no lock in the Account Status column, but there is a clock), and the account is assigned the HRPROFILE profile.
5. Minimize, but don't close, EM Express.

##### Create a User Account for Rachel Pandya by Using a Script

In this section, you create another user account named RPANDYA. Rather than use the EM Express interface to create this user, you use the SQL script that you generated in the previous section.

1. Open a new terminal window.
2. Change to the .ssh directory.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$

1. Use the SCP utility to copy the CreateHRUser.sql script you created in the previous practice to the compute node.

[oracle@edvm .ssh]$ **scp –i *your\_private\_key\_file* \**

* **/home/oracle/labs/CreateHRUser.sql \**

0.2KB/s 00:00

100% 182

CreateHRUser.sql

[oracle@edvm .ssh]$

* **oracle@ *your\_compute\_node\_IP\_Address*:/home/oracle/CreateHRUser.sql**

1. Connect to the compute node as the oracle user.

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

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

$ **sqlplus PDBADMIN/*password*@PDB1**

… SQL>

1. Execute the **CreateHRUser.sql** script. Enter **RPANDYA** when prompted for the username. Enter **HRCLERK** when prompted for the role. The order of the GRANT statements does not matter.

Grant succeeded.

Enter value for role: **HRCLERK**

old 1: grant &&role to &&username new 1: grant HRCLERK to RPANDYA

Grant succeeded.

SQL>

1: grant CREATE SESSION to &&username

1: grant CREATE SESSION to RPANDYA

old

new

SQL> **@/home/oracle/CreateHRUser.sql**

Enter value for username: **RPANDYA**

old 1: create user &&username identified by Pass4HR# new 1: create user RPANDYA identified by Pass4HR#

User created.

1. Return to EM Express and click the Reload current page button to refresh the data.
2. Scroll down the list and verify that the user RPANDYA has been created as expected.
3. Click **Log Out** and close the browser window.

##### Test DHAMBY's Access in SQL\*Plus

Connect to PDB1 as the DHAMBY user. Select the row with employee\_id=197 from the HR.EMPLOYEES table. Then attempt to delete it. You should get the “insufficient privileges” error. This happens because in Practice 9-5 Creating Local Roles in EM Express, you granted DHAMBY the HRCLERK role, which has SELECT and UPDATE privileges on the HR.EMPLOYEES table, not INSERT and DELETE.

No need to test for RPANDYA as this user has the same role as DHAMBY.

1. Return to the terminal window.
2. Connect to PDB1 as DHAMBY. When prompted, enter the new password. See Appendix— Product-Specific Credentials for the original and new password. When you enter the new password, it is not displayed in the interface.

SQL> **CONNECT DHAMBY/<password>@PDB1**

ERROR:

ORA-28001: the password has expired

Changing password for DHAMBY New password:

Retype new password:

Password changed Connected.

SQL>

1. View the salary for employee 197 from the HR.EMPLOYEES table. The query returns a value of 3000 for SALARY.

SQL> **SELECT salary FROM hr.employees WHERE employee\_id=197;**

SALARY

----------

3000

SQL>

1. Now attempt to delete the same row from the HR.EMPLOYEES table. DHAMBY is not allowed to perform DELETE operations on this table; therefore, the query returns an "insufficient privileges" error message.

SQL> **DELETE FROM hr.employees WHERE employee\_id=197;**

DELETE FROM hr.employees WHERE employee\_id=197

\* ERROR at line 1:

ORA-01031: insufficient privileges

SQL>

1. Disconnect from PDB1.

SQL> **DISCONNECT**

… SQL>

##### Test JGOODMAN's Access in SQL\*Plus

Repeat the test that you just did with DHAMBY with the JGOODMAN user account. After deleting the row, issue a ROLLBACK, so that you still have the original 107 rows.

1. Connect to PDB1 as JGOODMAN. Refer to *Course Practice Environment: Security Credentials* for the password value. When creating this user, you did not expire the password, so you won't have to change the password here.

SQL> **CONNECT JGOODMAN/*password*@PDB1**

Connected.

SQL>

1. Select the salary for employee 197 from the HR.EMPLOYEES table. The query returns a value of 3000 for SALARY.

SQL> **SELECT salary FROM hr.employees WHERE employee\_id=197;**

SALARY

----------

3000

SQL>

1. Delete the same row from the HR.EMPLOYEES table. JGOODMAN has the HRMANAGER role, and that role is granted SELECT, INSERT, UPDATE, and DELETE privileges on all tables in the HR schema. Therefore, the row is deleted.

SQL> **DELETE FROM hr.employees WHERE employee\_id=197;**

1 row deleted.

SQL>

1. Roll back the delete operation because this was just a test.

SQL> **ROLLBACK;**

Rollback complete.

SQL>

1. Confirm that you still have 107 rows in the HR.EMPLOYEES table.

SQL> **SELECT COUNT(\*) FROM hr.employees;**

COUNT(\*)

----------

107

SQL>

1. Disconnect from PDB1.

SQL> **DISCONNECT**

… SQL>

##### Test the Idle Time Limit in HRPROFILE

If you recall, in Practice 9-4 Creating a Local Profile in EM Express and Locking Accounts, you created a profile named HRPROFILE. In that profile, you configured the Idle Time limit to be 15 minutes. You assigned this profile to all three users (JGOODMAN, DHAMBY, and RPANDYA). In this section, you test that limit by connecting to PDB1 as RPANDYA and letting the session remain inactive for more than 15 minutes. After 15 minutes, verify that RPANDYA was automatically logged out by performing an operation; for example, try to select from the HR.EMPLOYEES table. While you're waiting, you can continue on to the next practice.

1. Connect to PDB1 as RPANDYA. When prompted, enter the new password. Refer to *Course Practice Environment: Security Credentials* for the password value. When you enter the new password, it is not displayed in the interface.

SQL> **CONNECT RPANDYA/*password*@PDB1**

ERROR:

ORA-28001: the password has expired

Changing password for RPANDYA New password:

Retype new password: Password changed Connected.

SQL>

1. Wait for 15 minutes. You can leave this terminal window open while waiting.
2. After 15 minutes, query the salary for employee 197 from the HR.EMPLOYEES table. The query returns the message "exceeded maximum idle time..." which indicates that HRPROFILE is working.

SQL> **SELECT salary FROM hr.employees WHERE employee\_id=197;**

ERROR at line 1:

ORA-02396: exceeded maximum idle time, please connect again SQL>

1. Exit SQL\*Plus.

## Practice 9-7: Configuring a Default Role for a User

### Overview

In this practice, PDBADMIN configures HRCLERK as the default role for JGOODMAN (user account for Jenny Goodman in PDB1). Jenny logs in to PDB1 and views the privileges that she gets from her default role. She requires more privileges to perform her management tasks, so she enables her non-default role, HRMANAGER, and views her new set of privileges.

### Assumptions

You are currently logged in as the oracle user.

You created the user account called JGOODMAN and granted it the HRMANAGER role, as well as the less-privileged HRCLERK role. To complete this practice, you must first complete the following practices:

* + Practice 9-3 Granting the DBA Role to PDBADMIN
  + Practice 9-4 Using EM Express to Create a Local Profile
  + Practice 9-5 Using EM Express to Create Local Roles
  + Practice 9-6 Using EM Express to Create Local Users

### Tasks

##### Configure a Default Role for JGOODMAN

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

$ **sqlplus PDBADMIN/*password*@PDB1**

…

SQL>

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

-------------------- --- HRCLERK YES

HRMANAGER YES

SQL>

DEF

GRANTED\_ROLE

SQL> **COLUMN granted\_role FORMAT A20**

SQL> **SELECT granted\_role, default\_role FROM dba\_role\_privs WHERE grantee='JGOODMAN';**

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

SQL> **ALTER USER JGOODMAN DEFAULT ROLE HRCLERK;**

User altered. SQL>

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

-------------------- --- HRCLERK YES

HRMANAGER NO

SQL>

DEF

GRANTED\_ROLE

SQL> **SELECT granted\_role, default\_role FROM dba\_role\_privs WHERE grantee='JGOODMAN';**

1. Disconnect PDBADMIN from PDB1.

SQL> **DISCONNECT**

…

SQL>

##### Enable a Non-Default Role

1. Connect to PDB1 as JGOODMAN.

SQL> **CONNECT JGOODMAN/*password*@PDB1**

Connected.

SQL>

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

SQL> **SELECT \* FROM session\_roles;**

ROLE

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

SQL>

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

SQL> **SET ROLE HRMANAGER;**

Role set.

SQL>

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

SQL> **SELECT \* FROM session\_roles;**

ROLE

---------------------------------------------------------------- HRMANAGER

SQL>

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

SQL> **SET ROLE HRMANAGER, HRCLERK;**

Role set.

SQL>

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

SQL> **SELECT \* FROM session\_roles;**

ROLE

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

HRMANAGER

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

## Practice 9-8: Exploring OS and Password File Authentication

### Overview

In this practice, you explore the OS and password file authentication.

### Assumptions

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

### Tasks

##### Exploring OS Authentication

During the course practices, you have logged in to the Oracle database as the oracle user and were authenticated using OS authentication. This section explores the groups and users in the Linux OS and how they are linked to authentication.

1. Linux and Unix operating systems have groups of users, and those are stored in the text file

/etc/group. Use the cat command to view the group file on the compute node. The format of each line is group\_name:password:Group ID (GID):user\_list. The Oracle Universal Installer creates the oinstall and dba groups in the OS. Notice that these groups are included in the list below. The dba group consists of the oracle user. The oinstall group does not have any users listed.

[oracle@MYDBCS ~]$ **cat /etc/group**

root:x:0: bin:x:1:bin,daemon daemon:x:2:bin,daemon

…

rpcuser:x:29: nfsnobody:x:65534: oinstall:x:54321: dba:x:54322:oracle

[oracle@MYDBCS ~]$

1. To find out the user that you are currently logged in as, execute whoami. The result shows that you are currently logged in to the OS as the oracle user.

[oracle@MYDBCS ~]$ **whoami**

oracle

[oracle@MYDBCS ~]$

1. Find out more about the oracle user. For example, verify that oracle is part of the dba

group.

* 1. The /etc/passwd file is a text file that lists user account information needed for logging in to the OS. Execute the following command to search for the oracle user. The format of the row is user:password:user ID:primary group ID:home

directory:shell that would run. Passwords are stored in the /etc/shadow file, so an x is used here instead.

[oracle@MYDBCS ~]$ **grep oracle /etc/passwd**

oracle:x:54321:54321::/home/oracle:/bin/bash

[oracle@MYDBCS ~]$

* 1. The information above tells you that oracle's primary group ID is 54321. To find the name of that group, search for it in the group file. The result shows that the oracle user's primary group is the oinstall group. However, a few steps back you saw that the oinstall group didn't have any users listed. This means that oracle is a user within a subgroup of oinstall.

[oracle@MYDBCS ~]$ **grep 54321 /etc/group**

oinstall:x:54321:

[oracle@MYDBCS ~]$

* 1. Investigate further. Search for oracle in the group file. The results tell you that oracle is a user in the dba group. The dba group is the Database Administrator Group, and any user in this group has the SYSDBA system privilege. So, if you log on to the Oracle database by using OS authentication and exercise the SYSDBA privilege, then the oracle user becomes the SYS user. If you recall, to log on using OS authentication, all you need to specify is CONNECT / AS SYSDBA. The / tells SQL\*Plus to look up the privileges for the OS user's group.

[oracle@MYDBCS ~]$ **grep oracle /etc/group**

dba:x:54322:oracle

[oracle@MYDBCS ~]$

##### Exploring Password Authentication

When you grant an administrative privilege to a user, for example, SYSDBA or SYSOPER, that user's name and privilege information are added to the database password file. The V$PWFILE\_USERS view contains information about users that have been granted administrative privileges.

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

privilege.

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

… SQL>

1. View the columns in the V$PWFILE\_USERS view by issuing the DESCRIBE command.

--------------------------- -------- --------------------------

USERNAME VARCHAR2(128)

Type

SQL> **DESCRIBE v$pwfile\_users**

Name Null?

SQL>

VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(30) VARCHAR2(128)

TIMESTAMP(9)WITH TIME ZONE DATE

DATE VARCHAR2(1024) VARCHAR2(8) VARCHAR2(3)

NUMBER

SYSDBA SYSOPER SYSASM SYSBACKUP SYSDG SYSKM

ACCOUNT\_STATUS PASSWORD\_PROFILE LAST\_LOGIN LOCK\_DATE EXPIRY\_DATE EXTERNAL\_NAME AUTHENTICATION\_TYPE COMMON

CON\_ID

1. List the users in the password file by querying the V$PWFILE\_USERS view.

SQL> **SELECT username FROM v$pwfile\_users;**

USERNAME

---------------------------------------------------------------- SYS

C##CDB\_ADMIN1

SQL>

1. Find out the SYS user's account status and whether the SYS user has the SYSDBA privilege by querying the V$PWFILE\_USERS view. ACCOUNT\_STATUS shows if the administrative user is OPEN, LOCKED (the user can no longer connect), or EXPIRED (the user must change the password at the connection).

------------------------------ ----- OPEN TRUE

SQL>

SYSDB

ACCOUNT\_STATUS

SQL> **SELECT account\_status, sysdba from v$pwfile\_users WHERE username='SYS';**

1. Exit SQL\*Plus and close the terminal window.

SQL> **EXIT**

**Practices for Lesson 10:**

**Creating PDBs**

## Practices for Lesson 10: Overview

### Overview

In these practices, you will create additional PDBs.

## Practice 10-1: Creating a PDB from Seed

### Overview

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

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

### Assumptions

You are logged in as the oracle user.

### Tasks

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Start SQL\*Plus and log in to your CDB with the SYSDBA privilege.

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

…

SQL>

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

SQL> **CREATE PLUGGABLE DATABASE PDB2**

1. **ADMIN USER PDB2ADMIN IDENTIFIED BY *password***
2. **ROLES=(dba)**;

Pluggable database created. SQL>

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

CREATE PLUGGABLE DATABASE PDB2

…

DEFAULT TABLESPACE USERS

DATAFILE '/u02/app/oracle/oradata/ORCL/PDB2/users01.dbf' SIZE 250M AUTOEXTEND ON

…

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

CREATE PLUGGABLE DATABASE PDB2

…

FILE\_NAME\_CONVERT=

('/u02/app/oracle/oradata/ORCL/pdbseed/', '/u02/app/oracle/oradata/ORCL/PDB2/', '/u04/app/oracle/oradata/temp/', '/u04/app/oracle/oradata/temp/PDB2/')

…

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

##### SQL> COLUMN con\_id FORMAT 999

##### SQL> COLUMN name FORMAT A10

##### SQL> SELECT con\_id, name, open\_mode FROM v$pdbs;

CON\_ID NAME OPEN\_MODE

------ ---------- ----------

* + 1. PDB$SEED READ ONLY
    2. PDB1 READ WRITE
    3. PDB2 MOUNTED

SQL>

* 1. Open PDB2 by using the ALTER PLUGGABLE DATABASE command.

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

Pluggable database altered.

SQL>

* 1. Verify that the open mode for PDB2 is now READ WRITE.

------ ---------- ----------

1. PDB$SEED READ ONLY
2. PDB1 READ WRITE
3. PDB2 READ WRITE

SQL>

OPEN\_MODE

CON\_ID NAME

SQL> **SELECT con\_id, name, open\_mode FROM v$pdbs;**

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

SQL> **!lsnrctl status**

LSNRCTL for Linux: Version 18.0.0.0.0 - Production on 22-MAR- 2018 16:00:59

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

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1521))) STATUS of the LISTENER

------------------------

Alias LISTENER

Version TNSLSNR for Linux: Version 18.0.0.0.0

- Production

Start Date 19-MAR-2018 15:23:07

Uptime 3 days 0 hr. 37 min. 52 sec

Trace Level off

Security ON: Local OS Authentication

SNMP OFF

Listener Parameter File

/u01/app/oracle/product/18.0.0/dbhome\_1/network/admin/listener.o ra

Listener Log File

/u01/app/oracle/diag/tnslsnr/MYDBCS/listener/alert/log.xml Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=1521))) (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=MYDBCS.compute-

588436052.oraclecloud.internal)(PORT=5500))(Security=(my\_wallet\_ directory=/u01/app/oracle/admin/ORCL/xdb\_wallet))(Presentation=H TTP)(Session=RAW))

Services Summary...

Service "66db11a937912ac9e0532618120ab4b9.588436052.oraclecloud.internal " has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "68035adc452a1241e0532618120ad62d.588436052.oraclecloud.internal " has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "ORCL.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "ORCL.588436052.oraclecloud.internalXDB" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "PDB1.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

Service "PDB2.588436052.oraclecloud.internal" has 1 instance(s).

Instance "ORCL", status READY, has 1 handler(s) for this service...

The command completed successfully

SQL>

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

SQL> **CONNECT**

**PDB2ADMIN/*password*@localhost:1521/PDB2.588436052.oraclecloud.int ernal**

Connected.

SQL>

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

command.

ALTER SESSION SET container = PDB2;

1. Explore PDB2.
   1. Show the current container.

SQL> **SHOW con\_name**

CON\_NAME

------------------------------ PDB2

SQL>

* 1. Show the current container ID.

SQL> **SHOW con\_id**

CON\_ID

------------------------------ 4

SQL>

* 1. List the service for PDB2 by querying the V$SERVICES view.

SQL> **COLUMN name FORMAT A20**

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

NAME

-------------------- PDB2

SQL>

* 1. List the data files for PDB2 and their respective tablespaces by querying the DBA\_DATA\_FILES view. Recall that DBCS uses OMF by default, so the files were created with the OMF file naming format.

------------------------------------------------------- ----------

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D SYSTEM 0A3C0D/datafile/o1\_mf\_system\_fk2t2tmq\_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D SYSAUX 0A3C0D/datafile/o1\_mf\_sysaux\_fk2t2tmv\_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D UNDOTBS1 0A3C0D/datafile/o1\_mf\_undotbs1\_fk2t2tmy\_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D USERS 0A3C0D/datafile/o1\_mf\_users\_fk2t2tn4\_.dbf

SQL>

TABLESPACE

FILE\_NAME

SQL> **col file\_name format a55**

SQL> **col tablespace\_name format a10**

SQL> **SELECT file\_name, tablespace\_name FROM dba\_data\_files;**

* 1. List the temp files for PDB2 by querying the DBA\_TEMP\_FILES view. The query returns one temp file. Your temp file name will be different from the one shown below.

SQL>

------------------------------------------------------- ----------

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D TEMP 0A3C0D/datafile/o1\_mf\_temp\_fk2t2tn2\_.dbf

TABLESPACE

FILE\_NAME

SQL> **SELECT file\_name, tablespace\_name FROM dba\_temp\_files;**

* 1. List the local users for PDB2 by querying the DBA\_USERS view.

SQL> **SELECT DISTINCT username FROM dba\_users WHERE common='NO' ORDER BY username;**

USERNAME

---------------------------------------------------------------- APEX\_050100

APEX\_INSTANCE\_ADMIN\_USER APEX\_LISTENER APEX\_PUBLIC\_USER APEX\_REST\_PUBLIC\_USER FLOWS\_FILES

PDB2ADMIN

7 rows selected.

SQL>

* 1. List the common users for PDB2 by querying the DBA\_USERS view.

SQL> **SELECT DISTINCT username FROM dba\_users WHERE common='YES' ORDER BY username;**

USERNAME

---------------------------------------------------------------- ANONYMOUS

APPQOSSYS AUDSYS C##CDB\_ADMIN1

… SYSTEM WMSYS

XDB

XS$NULL

39 rows selected.

SQL>

* 1. Exit SQL\*Plus.

SQL > **EXIT**

…

[oracle@MYDBCS ~]$

1. Add a service name entry to the tnsnames.ora file for PDB2.
   1. Change the directory to $ORACLE\_HOME/network/admin.

[oracle@MYDBCS ~]$ **cd $ORACLE\_HOME/network/admin**

[oracle@MYDBCS admin]$

* 1. View the tnsnames.ora file by using the cat command.

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

ORCL =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = ORCL.588436052.oraclecloud.internal)

)

)

PDB1 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = pdb1.588436052.oraclecloud.internal)

)

)

[oracle@MYDBCS admin]$

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

PDB2 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = pdb2.588436052.oraclecloud.internal)

)

)

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

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

ORCL =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = ORCL.588436052.oraclecloud.internal)

)

)

PDB1 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = pdb1.588436052.oraclecloud.internal)

)

)

PDB2 =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = MYDBCS.compute-

588436052.oraclecloud.internal)(PORT = 1521)) (CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = pdb2.588436052.oraclecloud.internal)

)

)

[oracle@MYDBCS admin]$

1. Connect to PDB2 by using the new service name and verify the current container.
   1. Start SQL\*Plus and connect to PDB2 as the SYSTEM user by using the PDB2 net service name.

[oracle@MYDBCS admin]$ **sqlplus system/*password*@PDB2**

…

SQL>

* 1. Verify that the current container name is PDB2.

SQL> **SHOW con\_name**

CON\_NAME

------------------------------ PDB2

SQL>

* 1. Exit SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS admin]$

## Practice 10-2: Cloning a PDB

### Overview

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

##### Tip

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

To set a title for a terminal window:

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

### Assumptions

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

### Tasks

##### Window 1: Create a Directory for PDB3

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Create a directory for the new PDB, named PDB3, under the CDB file location. You’ll first determine the correct location and then you’ll create the directory.
   1. Log in to SQL\*Plus as the SYS user with the SYSDBA privilege.

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

…

SQL>

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

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

NAME

----------------------------------------------------------------

/u02/app/oracle/oradata/ORCL/system01.dbf

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

/u02/app/oracle/oradata/ORCL/undotbs01.dbf

/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf

/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf

/u02/app/oracle/oradata/ORCL/users01.dbf

/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf

/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf

/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf

/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf

/u02/app/oracle/oradata/ORCL/PDB1/PDB1\_users01.dbf

/u02/app/oracle/oradata/ORCL/659622D851BF1AE2E0533620C40AD6D5/da tafile/o1\_mf\_users\_fjv8c7l1\_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da tafile/o1\_mf\_system\_fk2t2tmq\_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da tafile/o1\_mf\_sysaux\_fk2t2tmv\_.dbf

/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da tafile/o1\_mf\_undotbs1\_fk2t2tmy\_.dbf

16 rows selected.

SQL>

* 1. Use the host command to exit to the operating system.

SQL> **host**

[oracle@MYDBCS ~]$

* 1. Create a new directory named PDB3 in the location you determined in the previous step.

$ **mkdir /u02/app/oracle/oradata/ORCL/PDB3**

[oracle@MYDBCS ~]$

* 1. Enter exit to return to SQL\*Plus.

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

exit

SQL>

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

1. Switch to PDB1.

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

Session altered.

SQL>

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

--------------- -------------------------------- HR OPEN

SQL>

ACCOUNT\_STATUS

USERNAME

SQL> **col username format a15**

SQL> **SELECT username, account\_status FROM dba\_users where username = 'HR';**

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

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

Session altered.

SQL>

##### Window 2: Start a Transaction in PDB1

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

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle [oracle@MYDBCS ~]$

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

[oracle@MYDBCS ~]$ **sqlplus hr/*password*@PDB1**

…

SQL>

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

SQL> **SELECT salary FROM hr.employees WHERE employee\_id = 100;**

SALARY

----------

24000

SQL>

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

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

107 rows updated.

SQL>

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

SQL> **SELECT salary FROM hr.employees WHERE employee\_id = 100;**

SALARY

----------

26400

SQL>

##### Window 1: Clone PDB1 as PDB3

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

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

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

SQL> **CREATE PLUGGABLE DATABASE PDB3 FROM PDB1**

1. **CREATE\_FILE\_DEST='/u02/app/oracle/oradata/ORCL/PDB3'**
2. **KEYSTORE IDENTIFIED BY *keystore\_password*;**

Pluggable database created. SQL>

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

MOUNTED by querying the V$PDBS view.

------ ---------- ----------

2 PDB$SEED READ ONLY

4 PDB2 READ WRITE

3 PDB1 READ WRITE

5 PDB3 MOUNTED

SQL>

OPEN\_MODE

CON\_ID NAME

SQL> **COLUMN con\_id FORMAT 999**

SQL> **COLUMN name FORMAT A10**

SQL> **SELECT con\_id, name, open\_mode FROM v$pdbs;**

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

SQL> **ALTER PLUGGABLE DATABASE PDB3 OPEN;**

Pluggable database altered. SQL>

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

SQL>

READ WRITE NO

READ WRITE NO READ WRITE NO

NO

READ ONLY

2 PDB$SEED

4 PDB2

3 PDB1

5 PDB3

---------- ------------------------------ ---------- ----------

OPEN MODE RESTRICTED

SQL> **SHOW PDBS**

CON\_ID CON\_NAME

##### Window 2: Commit the Transaction

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

SQL> **COMMIT;**

Commit complete.

SQL>

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

SQL> **SELECT salary FROM hr.employees WHERE employee\_id = 100;**

SALARY

----------

26400

SQL>

1. Question: Do you think the salaries are updated in the clone (PDB3)? Answer: Continue to the next section to find out.

##### Window 1: Explore PDB3

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

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

Session altered.

SQL>

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

SQL> **SELECT salary FROM hr.employees WHERE employee\_id = 100;**

SALARY

----------

24000

SQL>

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

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

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

SQL> **COLUMN name FORMAT A20**

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

NAME

-------------------- PDB3

SQL>

1. List the data files for PDB3 and their respective tablespaces by querying the

DBA\_DATA\_FILES view. The results are formatted for easier viewing.

SQL> **SELECT file\_name, tablespace\_name FROM dba\_data\_files;**

FILE\_NAME

---------------------------------------------------------------- TABLESPACE\_NAME

------------------------------

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_system\_fcbkbm0d\_.dbf

SYSTEM

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_sysaux\_fcbkbm1o\_.dbf

SYSAUX

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_undotbs1\_fcbkbm1s\_.dbf

UNDOTBS1

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_users\_fcbkbm1z\_.dbf

USERS

SQL>

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

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

1. List the temp file(s) for PDB3 by querying the DBA\_TEMP\_FILES view. The query returns one temp file. The name of your temp file will be different from the one shown below.

SQL> **SELECT file\_name, tablespace\_name FROM dba\_temp\_files;**

FILE\_NAME

---------------------------------------------------------------- TABLESPACE\_NAME

------------------------------

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_temp\_fcbkbm1x\_.dbf

TEMP

SQL>

1. List the local users for PDB3 by querying the DBA\_USERS view.

SQL> **col username format a30**

SQL> **SELECT DISTINCT username FROM dba\_users WHERE common='NO' ORDER BY username;**

USERNAME

------------------------------ APEX\_050100

APEX\_INSTANCE\_ADMIN\_USER APEX\_LISTENER APEX\_PUBLIC\_USER APEX\_REST\_PUBLIC\_USER DHAMBY

FLOWS\_FILES HR INVENTORY JGOODMAN PDB1\_ADMIN1 PDBADMIN RPANDYA SCOTT

14 rows selected.

SQL>

1. List the common users for PDB3 by querying the DBA\_USERS view.

SQL> **SELECT DISTINCT username FROM dba\_users WHERE common='YES' ORDER BY username;**

USERNAME

------------------------------ ANONYMOUS

APPQOSSYS AUDSYS C##CDB\_ADMIN1

… SYSRAC SYSTEM WMSYS XDB

XS$NULL

39 rows selected.

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

##### Window 2: Return Salary Values to Their Original Values

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

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

107 rows updated.

SQL>

1. Commit the transaction.

SQL> **COMMIT;**

Commit complete.

SQL>

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

SQL> **SELECT salary FROM hr.employees WHERE employee\_id = 100;**

SALARY

----------

24000

SQL>

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

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

## Practice 10-3: Unplugging and Plugging in a PDB

### Overview

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

### Assumptions

You are logged in to the compute node as the oracle user. You completed Practice 10-2 Hot Cloning a PDB.

### Tasks

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

$ **sqlplus / as sysdba**

…

SQL>

* 1. Close PDB3.

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

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

Pluggable database altered.

SQL>

* 1. Unplug PDB3 into an XML file named /u02/app/oracle/oradata/PDB3.xml. The unplugging operation makes changes in the PDB data files to record that the PDB was properly and successfully unplugged.

Because the PDB is encrypted, you must include the ENCRYPT USING *transport\_secret* clause. If you do not include the clause, you will receive an ORA- 46680: master keys of the container database must be exported

error. Supply a value of TransPDB3 for *transport\_secret* for the course practice.

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

SQL> **ALTER PLUGGABLE DATABASE PDB3**

1. **UNPLUG INTO '/u02/app/oracle/oradata/PDB3.xml'**
2. **ENCRYPT USING TransPDB3;**

Pluggable database altered.

SQL>

* 1. Check the status of PDB3 by querying CDB\_PDBS.

---------- ---------- PDB2 NORMAL PDB$SEED NORMAL PDB1 NORMAL

PDB3 UNPLUGGED

SQL>

STATUS

PDB\_NAME

SQL> **col PDB\_NAME format a10**

SQL> **SELECT pdb\_name, status FROM cdb\_pdbs;**

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

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

Pluggable database dropped. SQL>

* 1. Verify the status of the unplugged PDB3 by querying the CDB\_PDBS view. Note that

PDB3 is not included.

---------- ---------- PDB2 NORMAL PDB$SEED NORMAL PDB1 NORMAL

SQL>

STATUS

PDB\_NAME

SQL> **SELECT pdb\_name, status FROM cdb\_pdbs;**

1. Plug PDB3 back into the ORCL CDB. The method would be similar if you were to plug the PDB into a different CDB.
   1. Make sure that PDB3 is compatible with the ORCL CDB. Execution of the following PL/SQL block raises an error if it is not compatible.

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

SQL> **set serveroutput on**

SQL> **DECLARE**

2 **compatible BOOLEAN := FALSE;**

**pdb\_descr\_file => '/u02/app/oracle/oradata/PDB3.xml');**

1. **if compatible then**
2. **DBMS\_OUTPUT.PUT\_LINE('PDB3 is compatible');**
3. **else DBMS\_OUTPUT.PUT\_LINE('PDB3 is not compatible');**
4. **end if;**
5. **END;**

10 **/**

PDB3 is compatible

PL/SQL procedure successfully completed. SQL>

**BEGIN**

**compatible := DBMS\_PDB.CHECK\_PLUG\_COMPATIBILITY(**

3

4

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

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

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

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

SQL> **CREATE PLUGGABLE DATABASE HRPDB**

1. **USING '/u02/app/oracle/oradata/PDB3.xml'**
2. **NOCOPY TEMPFILE REUSE**
3. **KEYSTORE IDENTIFIED BY *password***

4 **DECRYPT USING TransPDB3;**

Pluggable database created. SQL>

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

SQL> **COLUMN name FORMAT A8**

SQL> **SELECT name, con\_id FROM v$containers ORDER BY con\_id;**

|  |  |
| --- | --- |
| NAME | CON\_ID |
| -------- | ---------- |
| CDB$ROOT | 1 |
| PDB$SEED | 2 |
| PDB1 | 3 |
| PDB2 | 4 |
| HRPDB | 5 |
| SQL> |  |

* 1. Show the status of HRPDB by querying the CDB\_PDBS view.

---------- ---------- HRPDB NEW

SQL>

STATUS

PDB\_NAME

SQL> **SELECT pdb\_name, status FROM cdb\_pdbs WHERE pdb\_name='HRPDB';**

* 1. Show the open mode of HRPDB by querying the V$PDBS view.

SQL> **SELECT open\_mode FROM v$pdbs WHERE name='HRPDB';**

OPEN\_MODE

---------- MOUNTED

SQL>

* 1. List the data files of HRPDB by querying the V$DATAFILE view. Recall that the HRPDB

container's ID is 5. Your paths and data file names will differ from those shown below.

SQL> **COLUMN name FORMAT A50**

SQL> **SELECT name FROM v$datafile WHERE con\_id=5;**

NAME

----------------------------------------------------------------

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_system\_fcbkbm0d\_.dbf

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_sysaux\_fcbkbm1o\_.dbf

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_undotbs1\_fcbkbm1s\_.dbf

/u02/app/oracle/oradata/ORCL/PDB3/ORCL/68196836353470ABE053FA5E8 60AAA15/datafile/o1\_mf\_users\_fcbkbm1z\_.dbf

SQL>

1. Open and connect to HRPDB.
   1. Open HRPDB.

##### SQL> ALTER PLUGGABLE DATABASE HRPDB open;

Pluggable database altered. SQL>

* 1. Query V$SERVICES.

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

NAME

----------------------------------------------------------------

ORCL.588436052.oraclecloud.internal ORCL.588436052.oraclecloud.internalXDB SYS$BACKGROUND

SYS$USERS PDB1 PDB2

hrpdb

7 rows selected.

SQL>

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

SQL> **CONNECT**

**SYS/*password*@localhost:1521/hrpdb.588436052.oraclecloud.internal AS SYSDBA**

Connected.

SQL>

1. Verify the current container name is HRPDB.

SQL> **SHOW con\_name**

CON\_NAME

------------------------------ HRPDB

SQL>

1. Exit from SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS ~]$

## Practice 10-4: Dropping a PDB

### Overview

In this practice, you drop the HRPDB PDB.

### Assumptions

You are logged in to the compute node as the oracle user. You completed the following practices in this lesson:

* Creating a PDB from Seed
* Hot Cloning a PDB

### Tasks

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

privilege.

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

..

SQL>

1. List the PDBs in ORCL. The results show four PDBs: PDB$SEED, PDB1, PDB2, and HRPDB. SQL> **SHOW PDBS**

CON\_ID CON\_NAME OPEN MODE RESTRICTED

---------- ------------------------------ ---------- ----------

* 1. PDB$SEED READ ONLY NO
  2. PDB1 READ WRITE NO
  3. PDB2 READ WRITE NO
  4. HRPDB READ WRITE NO

SQL>

1. Close HRPDB.

SQL> **ALTER PLUGGABLE DATABASE HRPDB CLOSE;**

Pluggable database altered. SQL>

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

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

Pluggable database dropped. SQL>

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

SQL>

READ WRITE NO

READ WRITE NO

NO

READ ONLY

1. PDB$SEED
2. PDB1
3. PDB2

---------- ------------------------------ ---------- ----------

OPEN MODE RESTRICTED

SQL> **SHOW PDBS**

CON\_ID CON\_NAME

1. Exit SQL\*Plus and close the terminal window.

SQL> **EXIT**

…

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

# Practices for Lesson 11: Creating Master Encryption Keys for PDBs

## Practices for Lesson 11: Overview

### Overview

In these practices, you will create and activate an encryption key.

## Practice 11-1: Creating and Activating an Encryption Key

### Overview

In this practice, you create a new PDB named PDBTEST. You then determine whether you need to create and activate an encryption key for the new PDB. If needed, you create and activate an encryption key.

### Assumptions Tasks

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

###### oracle@your\_compute\_node\_IP\_Address

[oracle@MYDBCS ~]$

1. Use oraenv to set ORACLE\_SID to ORCL.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle [oracle@MYDBCS ~]$

1. Start SQL\*Plus and log in to your CDB with the SYSDBA privilege.

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

… SQL>

1. Create PDBTEST by using the CREATE PLUGGABLE DATABASE command. Specify an admin user named PDBTESTADMIN and grant this user the DBA role.

##### SQL> CREATE PLUGGABLE DATABASE pdbtest

##### ADMIN USER pdbtestadmin IDENTIFIED BY DBAdmin\_1

* 1. **ROLES=(dba)**; Pluggable database created.

SQL>

1. Open PDBTEST by using the ALTER PLUGGABLE DATABASE command.

SQL> **ALTER PLUGGABLE DATABASE PDBTEST OPEN;**

Pluggable database altered. SQL>

1. Verify that the open mode for PDBTEST is now READ WRITE. Also, note the container ID for

PDBTEST.

---------- --------------- ----------

1. PDB$SEED READ ONLY
2. PDB1 READ WRITE
3. PDB2 READ WRITE
4. PDBTEST READ WRITE

SQL>

OPEN\_MODE

CON\_ID NAME

SQL> **col name format a15**

SQL> **SELECT con\_id, name, open\_mode FROM v$pdbs;**

1. Set the container to PDBTEST.

SQL> **ALTER SESSION SET CONTAINER = pdbtest;**

Session altered.

SQL>

1. Query the STATUS column in V$ENCRYPTION\_WALLET. If STATUS contains a value of

OPEN\_NO\_MASTER\_KEY, you must create and activate the master encryption key.

------------------------------ -------------------- OPEN\_NO\_MASTER\_KEY AUTOLOGIN

SQL>

WALLET\_TYPE

STATUS

SQL> **SELECT status, wallet\_type FROM v$encryption\_wallet;**

1. Create and activate a master encryption key in the PDB by executing the following command. Refer to *Course Practice Environment: Security Credentials* for the password value.

SQL> **ADMINISTER KEY MANAGEMENT SET KEY FORCE KEYSTORE**

1. **IDENTIFIED BY *keystore-password***
2. **WITH BACKUP USING 'pdbtest\_bkup';**

keystore altered. SQL>

1. Query V$ENCRYPTION\_WALLET again to verify that the value in the STATUS column OPEN.

------------------------------ -------------------- OPEN AUTOLOGIN

SQL>

WALLET\_TYPE

STATUS

SQL> **SELECT status, wallet\_type FROM v$encryption\_wallet;**

1. You will not need this PDB for any other practices, so you can now drop it.
2. Set the container to the root.

##### SQL> ALTER SESSION set container = cdb$root;

Session altered.

1. Close PDBTEST.

##### SQL> ALTER PLUGGABLE DATABASE PDBTEST CLOSE;

Pluggable database altered.

SQL>

1. Drop PDBTEST.

SQL> **DROP PLUGGABLE DATABASE PDBTEST INCLUDING DATAFILES;**

Pluggable database dropped. SQL>

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

SQL>

READ WRITE NO

READ WRITE NO

NO

READ ONLY

1. PDB$SEED
2. PDB1
3. PDB2

---------- ------------------------------ ---------- ----------

OPEN MODE RESTRICTED

SQL> **SHOW PDBS**

CON\_ID CON\_NAME

1. Exit SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS ~]$

## Practice 11-2: Creating and Activating the Encryption Key for PDB2

### Overview

In this practice, you determine whether you need to create and activate the master encryption key for PDB2.

### Assumptions

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

### Tasks

1. Start SQL\*Plus and log in to your CDB with the SYSDBA privilege.

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

…

SQL>

1. Verify that the open mode for PDB2 is READ WRITE.

---------- --------------- ----------

1. PDB$SEED READ ONLY
2. PDB1 READ WRITE
3. PDB2 READ WRITE

SQL>

OPEN\_MODE

CON\_ID NAME

SQL> **col name format a15**

SQL> **SELECT con\_id, name, open\_mode FROM v$pdbs;**

1. Set the container to PDB2.

SQL> **ALTER SESSION SET CONTAINER = pdb2;**

Session altered.

SQL>

1. Query the STATUS column in V$ENCRYPTION\_WALLET. If STATUS contains a value of

OPEN\_NO\_MASTER\_KEY, you must create and activate the master encryption key.

------------------------------ -------------------- OPEN\_NO\_MASTER\_KEY AUTOLOGIN

SQL>

WALLET\_TYPE

STATUS

SQL> **SELECT status, wallet\_type FROM v$encryption\_wallet;**

1. Create and activate the master encryption key in PDB2. Refer to *Course Practice Environment: Security Credentials* for the password value.

SQL> **ADMINISTER KEY MANAGEMENT SET KEY FORCE KEYSTORE**

1. **IDENTIFIED BY *keystore-password***
2. **WITH BACKUP USING 'pdb2\_bkup';**

keystore altered. SQL>

1. Query V$ENCRYPTION\_WALLET again to verify that the STATUS column is set to OPEN.

------------------------------ -------------------- OPEN AUTOLOGIN

SQL>

WALLET\_TYPE

STATUS

SQL> **SELECT status, wallet\_type FROM v$encryption\_wallet;**

1. Exit SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS ~]$

**Practices for Lesson 12: Creating and Managing**

**Tablespaces**

## Practices for Lesson 12: Overview

### Overview

In these practices, you will view information about tablespaces and create new tablespaces.

## Practice 12-1: Viewing Tablespace Information

### Overview

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

### Assumptions

You are logged in as the oracle user.

### Tasks

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

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

[oracle@MYDBCS ~]$ **sqlplus PDBADMIN/*password*@PDB1**

…

SQL>

1. List the columns in the DBA\_TABLESPACES view by using the DESCRIBE command.

VARCHAR2(14) VARCHAR2(12) VARCHAR2(1000)

VARCHAR2(7)

NOT NULL VARCHAR2(30) NOT NULL NUMBER

NUMBER NUMBER

NOT NULL NUMBER

TABLESPACE\_NAME BLOCK\_SIZE INITIAL\_EXTENT NEXT\_EXTENT MIN\_EXTENTS

…

DEF\_CELLMEMORY DEF\_INMEMORY\_SERVICE DEF\_INMEMORY\_SERVICE\_NAME

LOST\_WRITE\_PROTECT

----------------------------------------- -------- ------------

Type

Null?

SQL> **DESCRIBE dba\_tablespaces**

Name

SQL>

VARCHAR2(1)

CHUNK\_TABLESPACE

1. List the tablespaces in PDB1.

SQL> **SELECT DISTINCT tablespace\_name FROM dba\_tablespaces ORDER BY tablespace\_name;**

TABLESPACE\_NAME

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

SYSTEM TEMP UNDOTBS1 USERS

SQL>

1. Find out which tablespace contains the HR schema by querying the ALL\_TABLES view.

##### SQL> SELECT DISTINCT tablespace\_name FROM all\_tables WHERE owner='HR';

TABLESPACE\_NAME

------------------------------ USERS

SQL>

1. Query the STATUS, CONTENTS, LOGGING, PLUGGED\_IN, BIGFILE, EXTENT\_MANAGEMENT, and ALLOCATION\_TYPE columns in the DBA\_TABLESPACES view for the SYSAUX tablespace.

--------- ------------------ --------- --- --- ---------- ------ ONLINE PERMANENT LOGGING NO NO LOCAL SYSTEM

SQL>

LOGGING PLU BIG EXTENT\_MAN ALLOCA

CONTENTS

STATUS

SQL> **SELECT status, contents, logging, plugged\_in, bigfile, extent\_management, allocation\_type FROM dba\_tablespaces where tablespace\_name='SYSAUX';**

* + STATUS shows the value ONLINE, indicating the tablespace is available to users.
  + CONTENTS indicates the PERMANENT tablespace type.
  + LOGGING shows the value LOGGING, indicating that certain DML operations are logged in the redo log file.
  + PLUGGED\_IN shows the value NO, indicating that the tablespace is not plugged in.
  + BIGFILE shows the value NO, indicating that the tablespace is a smallfile tablespace.
  + EXTENT\_MANAGEMENT shows the value LOCAL, indicating that the tablespace is locally managed (not dictionary managed).
  + ALLOCATION\_TYPE shows the value SYSTEM, indicating that the extents of the tablespace are managed by the system, and you cannot specify an extent size.

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

----------------------------------------- -------- ------------ TS# NUMBER

NAME VARCHAR2(30)

INCLUDED\_IN\_DATABASE\_BACKUP VARCHAR2(3)

BIGFILE VARCHAR2(3)

FLASHBACK\_ON VARCHAR2(3)

ENCRYPT\_IN\_BACKUP VARCHAR2(3)

CON\_ID NUMBER

SQL>

Type

Null?

SQL> **DESCRIBE v$tablespace**

Name

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

---------- ------------------------------ --- --- --- --- ------

1 SYSAUX YES NO YES 3

SQL>

INC BIG FLA ENC CON\_ID

TS# NAME

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

* + INCLUDED\_IN\_DATABASE\_BACKUP contains the value YES, indicating that the

tablespace is included in full database backups by using the BACKUP DATABASE

RMAN command.

* + BIGFILE contains the value NO, indicating that the tablespace is a smallfile tablespace.
  + FLASHBACK\_ON contains the value YES, indicating that the tablespace participates in

FLASHBACK DATABASE operations.

* + ENCRYPT\_IN\_BACKUP contains the value null, indicating that encryption is neither explicitly turned on nor off at the tablespace level.
  + CON\_ID indicates the container to which the data pertains. In this case, PDB1 is container ID 3.

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

SQL> **SELECT table\_name FROM all\_tables WHERE tablespace\_name='USERS' and owner='HR';**

TABLE\_NAME

---------------------------------------------------------------- REGIONS

LOCATIONS DEPARTMENTS JOBS EMPLOYEES JOB\_HISTORY

6 rows selected.

SQL>

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

SQL> **SELECT index\_name FROM all\_indexes WHERE tablespace\_name='USERS' AND owner='HR' ORDER BY index\_name;**

INDEX\_NAME

---------------------------------------------------------------- COUNTRY\_C\_ID\_PK

DEPT\_ID\_PK

DEPT\_LOCATION\_IX EMP\_DEPARTMENT\_IX EMP\_EMAIL\_UK EMP\_EMP\_ID\_PK EMP\_JOB\_IX EMP\_MANAGER\_IX EMP\_NAME\_IX JHIST\_DEPARTMENT\_IX JHIST\_EMPLOYEE\_IX JHIST\_EMP\_ID\_ST\_DATE\_PK JHIST\_JOB\_IX

JOB\_ID\_PK LOC\_CITY\_IX LOC\_COUNTRY\_IX LOC\_ID\_PK

LOC\_STATE\_PROVINCE\_IX

REG\_ID\_PK

19 rows selected.

SQL>

1. List the columns in the DBA\_DATA\_FILES view by using the DESCRIBE command. You can query this view to learn about the data files contained in a tablespace.

----------------------------------------- -------- ------------ FILE\_NAME VARCHAR2(513)

FILE\_ID NUMBER

TABLESPACE\_NAME VARCHAR2(30)

BYTES NUMBER

BLOCKS NUMBER

STATUS VARCHAR2(9)

RELATIVE\_FNO NUMBER

AUTOEXTENSIBLE VARCHAR2(3)

MAXBYTES NUMBER

MAXBLOCKS NUMBER

INCREMENT\_BY NUMBER

USER\_BYTES NUMBER

USER\_BLOCKS NUMBER

ONLINE\_STATUS VARCHAR2(7)

LOST\_WRITE\_PROTECT VARCHAR2(7)

SQL>

Type

Null?

SQL> **DESCRIBE dba\_data\_files**

Name

1. List data file information for the SYSAUX tablespace by querying various columns in the

DBA\_DATA\_FILES view.

SQL>

BYTES MAXBYTES USER\_BYTES

-------------------------------------------------- --- ---------

/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf YES 828375040

3.4360E+10 827326464

AUT

FILE\_NAME

SQL> **COLUMN file\_name FORMAT A50**

SQL> **SELECT file\_name, autoextensible, bytes, maxbytes, user\_bytes FROM dba\_data\_files WHERE tablespace\_name='SYSAUX';**

The results show the following:

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

1. Find out how many segments are there in the SYSAUX tablespace by querying the

DBA\_SEGMENTS view.

SQL> **SELECT count(segment\_name) FROM dba\_segments WHERE tablespace\_name='SYSAUX';**

COUNT(SEGMENT\_NAME)

-------------------

2356

SQL>

1. Find out which index in the SYSAUX tablespace takes up the most space by querying the DBA\_SEGMENTS view. The results indicate that the I\_WRI$\_OPTSTAT\_H\_OBJ#\_ICOL#\_ST index takes up the most space.

----------------------------------- ------------------ --------- I\_WRI$\_OPTSTAT\_H\_OBJ#\_ICOL#\_ST INDEX 42991616

SQL>

BYTES

SEGMENT\_TYPE

SEGMENT\_NAME

SQL> **col segment\_name format a35**

SQL> **SELECT \***

1. **FROM (SELECT segment\_name, segment\_type, bytes**
2. **FROM dba\_segments**
3. **WHERE segment\_type = 'INDEX' AND**
4. **tablespace\_name ='SYSAUX'**
5. **ORDER BY bytes desc)**
6. **WHERE rownum < 2;**
7. Exit SQL\*Plus.

SQL> **exit**

..

[oracle@MYDBCS ~]$

##### Viewing Tablespace Information by Using Enterprise Manager Database Express

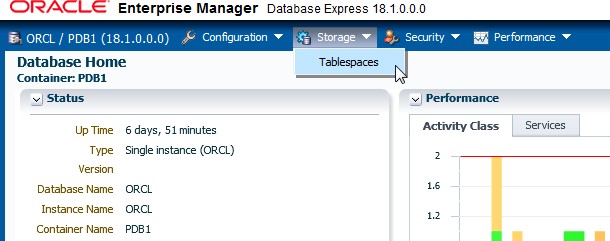
1. Create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
   1. Open a new terminal window.
   2. Use ssh to create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

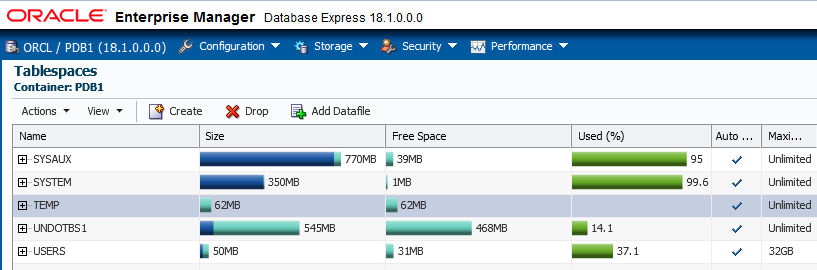
[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

[oracle@MYDBCS ~]$

1. Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. On the Enterprise Manager Database Express login page, enter the following:
   1. In the User Name field, enter **PDBADMIN**.
   2. In the Password field, enter the password you specified when you created the database deployment.
   3. In the Container Name field, enter **PDB1**.
3. Click Login.
4. Expand **Storage** and then select **Tablespaces**.



1. All the tablespaces are listed with their size, amount of free space, amount used (%), Auto Extend setting, Maximum Size setting, Status, Type, Group Name, Auto Segment Management setting, and Directory.



1. Question: In this example, how much of the SYSAUX tablespace is used?

Answer: 95% of the SYSAUX tablespace has been used. It has 39MB of free space left.

**Note:** The values in your database may differ from what is shown in this example.

1. Log out of EM Express.

## Practice 12-2: Creating a Tablespace

### Overview

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

### Assumptions

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

### Tasks

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

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

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

[oracle@MYDBCS ~]$ **sqlplus PDBADMIN/*password*@PDB1**

… SQL>

1. Execute the CreateINVENTORYTablespace.sql script.

SQL> **set echo on**

SQL> **@/home/oracle/labs/CreateINVENTORYTablespace.sql**

SQL> CREATE SMALLFILE TABLESPACE INVENTORY

2 DATAFILE 3

'/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF' SIZE 5242880

1. DEFAULT NOCOMPRESS
2. ONLINE
3. SEGMENT SPACE MANAGEMENT AUTO
4. EXTENT MANAGEMENT LOCAL AUTOALLOCATE; Tablespace created.

SQL>

1. Execute the CreateTable\_X.sql script to create and populate the X table. Notice that near the end, you get an error message: unable to extend table PDBADMIN.X by

128 in tablespace INVENTORY. You get this message because the tablespace in which you are trying to create table X is too small. You will remedy this problem in the next section.

##### SQL> @/home/oracle/labs/CreateTable\_X.sql

PL/SQL procedure successfully completed.

SQL> CREATE TABLE x 2 (a CHAR(1000)

3 ) TABLESPACE inventory;

Table created.

SQL> INSERT INTO x

2 VALUES ('a');

1 row created.

SQL> INSERT INTO x

2 SELECT \* FROM x;

1 row created.

…

SQL> INSERT INTO x

2 SELECT \* FROM x ;

1024 rows created.

SQL> INSERT INTO x

2 SELECT \* FROM x ; INSERT INTO x

\*

ERROR at line 1:

ORA-01653: unable to extend table PDBADMIN.X by 128 in tablespace INVENTORY

SQL> COMMIT;

Commit complete.

SQL> quit

…

[oracle@MYDBCS ~]$

##### Use EM Database Express to Increase the Size of the INVENTORY01.DBF Data File

Fix the problem that you encountered in the previous section by increasing the size of the INVENTORY01.dbf data file. Use EM Express because it provides an easy-to-use interface when working with tablespaces.

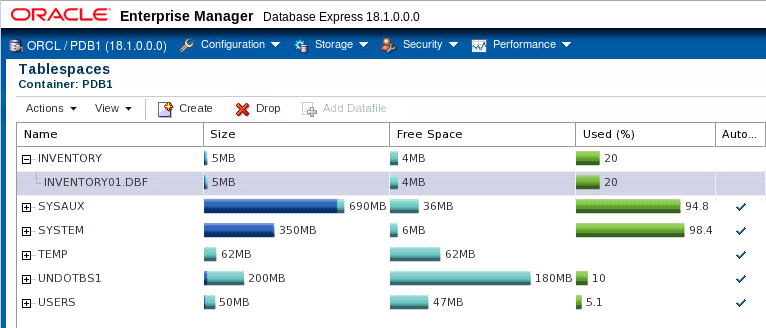
1. Create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
   1. Open a terminal window.
   2. Use ssh to create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

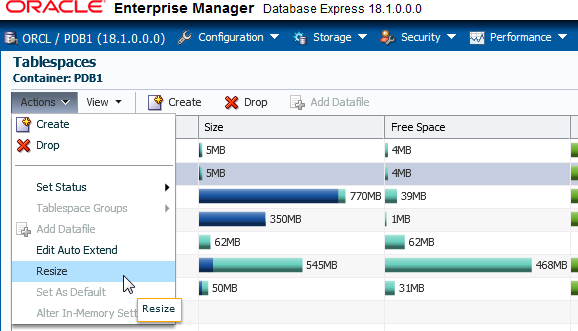
[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

[oracle@MYDBCS ~]$

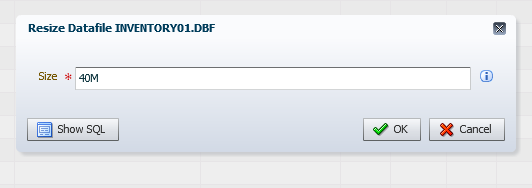
1. On your Linux desktop, open Firefox. Launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. On the Enterprise Manager Database Express login page, enter the following:
   1. In the User Name field, enter **PDBADMIN**.
   2. In the Password field, enter the password you specified when you created the database deployment.
   3. In the Container Name field, enter **PDB1**.
3. Click **Login**.
4. Expand **Storage** and then select **Tablespaces**.
5. Expand the **INVENTORY** tablespace and select the **INVENTORY01.DBF** data file.



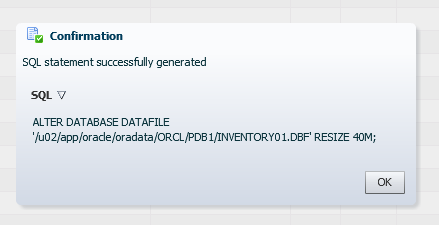
1. Expand **Actions** and then select **Resize**.



1. The "Resize Datafile INVENTORY01.DBF" window is displayed. In the Size box, enter **40M**. Don't click OK just yet.

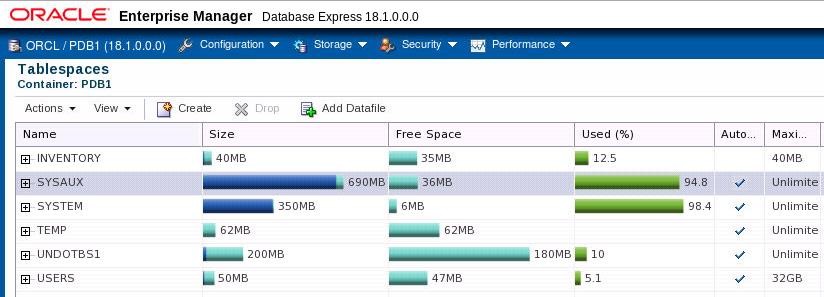


1. Click the **Show SQL** button to view the SQL command that performs the resize action.
2. In the Confirmation dialog box, click **OK**.



1. In the "Resize Datafile INVENTORY01.DBF" dialog box, click **OK**. The SQL code is executed, and the data file is successfully resized.
2. In the Confirmation dialog box, click **OK**.
3. Verify that the change is reflected in the EM Express interface. The size for the INVENTORY

tablespace should now be set to 40MB.

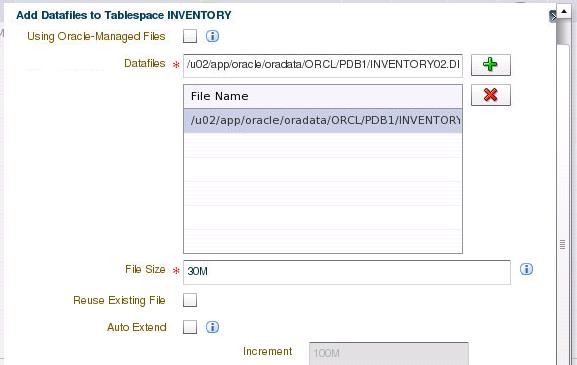


##### Use EM Database Express to Add a Data File to the INVENTORY Tablespace

1. Select the **INVENTORY** tablespace.
2. Expand **Actions** and then select **Add Datafile**.
3. The "Add Datafiles to Tablespace INVENTORY" dialog box is displayed.
   1. Deselect **Using Oracle-Managed Files**.
   2. In the Datafiles field, enter

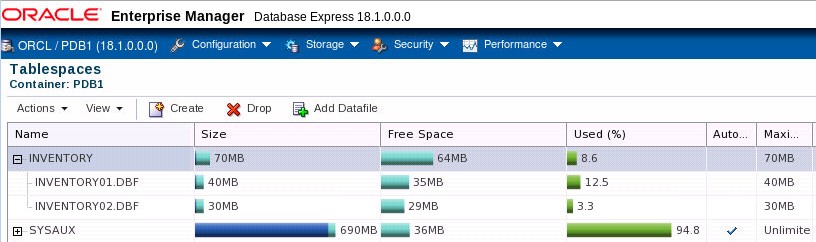
**/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY02.DBF** and click the plus sign to add the file to the list.

* 1. In the File Size field, enter **30M**.
  2. Deselect **Auto Extend**.



1. Click **Show SQL** and view the SQL code being generated.
2. A Confirmation dialog box displays the code. In the Confirmation dialog box, click **OK**.
3. In the "Add Datafiles to Tablespace INVENTORY" window, click **OK**. A message states that the data file was successfully added to the INVENTORY tablespace.
4. In the Confirmation dialog box, click **OK**.
5. Expand the **INVENTORY** tablespace and verify that it now has two data files:

##### INVENTORY01.DBF and INVENTORY02.DBF.



1. Click Log Out.

##### Use SQL\*Plus to Create Table X and Populate It

As the PDBADMIN user, run the script named CreateTableX.sql again in SQL\*Plus to create and populate the table called X in the INVENTORY tablespace. This time you shouldn't receive an error because you increased the size of the tablespace.

1. Return to your terminal window.
2. Start SQL\*Plus and connect to PDB1 as the PDBADMIN user.

$ **sqlplus PDBADMIN/*password*@PDB1**

…

SQL>

1. Run the CreateTable\_X.sql script, located in /home/oracle/labs. The script runs without any errors.

SQL> **@/home/oracle/labs/CreateTable\_X.sql**

PL/SQL procedure successfully completed. SQL> CREATE TABLE x

2 (a CHAR(1000)

3 ) TABLESPACE inventory; Table created.

SQL> INSERT INTO x

2 VALUES ('a');

1 row created.

SQL> INSERT INTO x

2 SELECT \* FROM x;

1 row created.

…

SQL> INSERT INTO x

2 SELECT \* FROM x ; 2048 rows created. SQL> COMMIT;

Commit complete.

SQL> quit

…

[oracle@MYDBCS ~]$

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

$ **sqlplus PDBADMIN/*password*@PDB1**

… SQL>

1. Verify that table X was created in the INVENTORY tablespace.

SQL> **SELECT table\_name FROM all\_tables WHERE tablespace\_name='INVENTORY';**

TABLE\_NAME

---------------------------------------------------------------- X

SQL>

##### Use SQL\*Plus to Drop the INVENTORY Tablespace

1. Drop the INVENTORY tablespace.

SQL> **DROP TABLESPACE inventory INCLUDING CONTENTS AND DATAFILES;**

Tablespace dropped. SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS ~]$

## Practice 12-3: Creating a Tablespace that is Encrypted by Default

### Overview

In this practice, you determine whether tablespace encryption by default is configured. You then create a new tablespace and verify that it is encrypted by default.

### Tasks

1. Log in to SQL\*Plus as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus system/*password***

…

SQL>

1. Use the SHOW PARAMETER command to verify that tablespace encryption by default is configured.

CLOUD\_ONLY

string

encrypt\_new\_tablespaces

SQL>

------------------------------------ ----------- ---------------

VALUE

TYPE

SQL> **SHOW PARAMETER encrypt**

NAME

1. Set the container to PDB1.

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

Session altered.

SQL>

1. Check whether any of the tablespaces in PDB1 are encrypted.

------------------------------ --- SYSTEM NO

SYSAUX NO

UNDOTBS1 NO

TEMP NO

USERS YES

SQL>

ENC

TABLESPACE\_NAME

SQL> **SELECT tablespace\_name, encrypted FROM dba\_tablespaces;**

1. Question: Why is the USERS tablespace encrypted?

Answer: The ENCRYPTION\_NEW\_TABLESPACES initialization parameter is set to

CLOUD\_ONLY and the tablespace is a non-system tablespace.

1. Create a new tablespace named TESTENCRYPT in PDB1.

SQL> **CREATE TABLESPACE testencrypt DATAFILE**

**'/u02/app/oracle/oradata/ORCL/PDB1/testencrypt01.dbf' SIZE 20M;**

Tablespace created. SQL>

1. Question: Is the new tablespace encrypted?

------------------------------ --- SYSTEM NO

SYSAUX NO

UNDOTBS1 NO

TEMP NO

USERS YES

TESTENCRYPT YES

6 rows selected.

SQL>

ENC

TABLESPACE\_NAME

SQL> **SELECT tablespace\_name, encrypted FROM dba\_tablespaces;**

Answer: Yes.

1. Drop the TESTENCRYPT tablespace.

SQL> **DROP TABLESPACE testencrypt INCLUDING CONTENTS AND DATAFILES;**

Tablespace dropped. SQL>

1. Exit SQL\*Plus and close the terminal window.

SQL> **exit**

…

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

**Practices for Lesson 13:**

**Managing Storage Space**

## Practices for Lesson 13: Overview

### Overview

In In these practices, you will use the Segment Advisor to manage space in your database. You will also use the Compression Advisor. Finally, you enable the Resumable Space Allocation feature.

## Practice 13-1: Managing Space in Tablespaces

### Overview

In this practice, you will set a warning threshold and a critical threshold on a tablespace and then test those thresholds. You then create a Segment Advisor task to get recommendations about the current space situation.

##### Tip

For problems that cannot be resolved automatically and require DBAs to be notified, such as running out of space, the Oracle Database server provides server-generated alerts. Two alert thresholds are defined by default:

* + The warning threshold is the limit at which space is beginning to run low.
  + The critical threshold is a serious limit that warrants your immediate attention.

The database issues alerts at both thresholds. The alerts notify you and often provide recommendations on how to resolve the reported problem.

### Tasks

##### Set a Warning Threshold

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Connect to **PDB1** as the **SYSTEM** user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus SYSTEM/*password*@PDB1**

…

SQL>

1. Execute the DBMS\_SERVER\_ALERT.SET\_THRESHOLD procedure to reset the database- wide threshold values for the Tablespace Space Usage metric.

SQL> **exec DBMS\_SERVER\_ALERT.SET\_THRESHOLD(-**

* **dbms\_server\_alert.tablespace\_pct\_full,-**
* **NULL,NULL,NULL,NULL,1,1,NULL,-**
* **dbms\_server\_alert.object\_type\_tablespace,NULL);**

PL/SQL procedure successfully completed.

SQL>

1. Check the database-wide threshold values for the Tablespace Space Usage metric.
   1. Connect to the root container.

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

Session altered.

SQL>

* 1. Query the WARNING\_VALUE and the CRITICAL\_VALUE columns in the DBA\_THRESHOLDS view. The results show that the warning threshold value is 85 and the critical threshold value is 97.

-------------------- -------------------- 85 97

SQL>

CRITICAL\_VALUE

WARNING\_VALUE

SQL> **col warning\_value format a20**

SQL> **col critical\_value format a20**

SQL> **SELECT warning\_value, critical\_value FROM dba\_thresholds WHERE metrics\_name='Tablespace Space Usage' AND object\_name IS NULL;**

1. In PDB1, create a new tablespace called TBSALERT with a 120MB file called tbsalert.dbf. Make sure that this tablespace is locally managed and uses Automatic Segment Space Management. Do not make it auto-extensible and do not specify any thresholds for this tablespace.
   1. Connect to PDB1.

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

Session altered.

SQL>

* 1. Create the TBSALERT tablespace by executing the Create\_TBSALERT\_TS.sql

script.

SQL> **set echo on**

SQL> **@/home/oracle/labs/Create\_TBSALERT\_TS.sql**

SQL> CREATE TABLESPACE tbsalert

1. DATAFILE '/u02/app/oracle/oradata/ORCL/PDB1/tbsalert.dbf'
2. SIZE 120M REUSE LOGGING EXTENT MANAGEMENT LOCAL

4 SEGMENT SPACE MANAGEMENT AUTO; Tablespace created.

SQL>

1. Query how much free space the TBSALERT tablespace holds by executing the

$HOME/labs/TBSALERT\_free\_space.sql script.

------------------------------ ---------- ---------- ---------- TBSALERT 124780544 125829120 99.1666667

SQL>

BYTES PCT\_FREE

FREE

TABLESPACE

df.tablespace\_name = 'TBSALERT';

AND

df.bytes , fs.bytes \*100/ df.bytes PCT\_FREE

FROM dba\_data\_files df ,dba\_free\_space fs WHERE df.tablespace\_name = fs.tablespace\_name

2

3

4

5

SQL> **set echo on**

SQL> **@/home/oracle/labs/TBSALERT\_free\_space.sql**

SQL> SELECT df.tablespace\_name tablespace, fs.bytes free,

1. Modify the thresholds values for the Tablespace Space Usage metric for the TBSALERT

tablespace. Set the Warning Threshold to 55 and the Critical Threshold to 70.

SQL> **exec DBMS\_SERVER\_ALERT.SET\_THRESHOLD( metrics\_id => dbms\_server\_alert.tablespace\_pct\_full, warning\_operator => DBMS\_SERVER\_ALERT.OPERATOR\_GE, warning\_value => '55', critical\_operator => DBMS\_SERVER\_ALERT.OPERATOR\_GE, critical\_value => '70', observation\_period => 1, consecutive\_occurrences => 1, instance\_name => 'ORCL', object\_type => DBMS\_SERVER\_ALERT.OBJECT\_TYPE\_TABLESPACE,**

**object\_name => 'TBSALERT')**

PL/SQL procedure successfully completed. SQL>

1. Verify that the thresholds are set correctly. The query returns a warning value of 55 and a critical value of 70, which indicates that the thresholds are set correctly.

-------------------- --------------------

55 70

CRITICAL\_VALUE

WARNING\_VALUE

SQL> **SELECT warning\_value, critical\_value FROM dba\_thresholds WHERE object\_name='TBSALERT';**

SQL>

1. Query the REASON and RESOLUTION columns from the DBA\_ALERT\_HISTORY view for the

TBSALERT tablespace.

-------------------------------------------------------- -------

Threshold is updated on metrics "Tablespace Space Usage" cleared

SQL>

RESOLUT

REASON

SQL> **col reason format a60**

SQL> **SELECT reason, resolution FROM dba\_alert\_history WHERE object\_name='TBSALERT';**

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Execute the $HOME/labs/seg\_advsr\_setup.sh shell script to create and populate new tables in the TBSALERT tablespace.

$ **$HOME/labs/seg\_advsr\_setup.sh**

SQL\*Plus: Release 18.0.0.0.0 Production on Thu Mar 29 18:24:59

2018

Version 18.1.0.0.0

Copyright (c) 1982, 2017, Oracle. All rights reserved. SQL> Connected.

SQL>

System altered.

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

SQL> ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes Variable Size 704643072 bytes

Database Buffers 1979711488 bytes

Redo Buffers 74985472 bytes Database mounted.

PL/SQL procedure successfully completed.

SQL>

109568 rows created.

SQL>

109568 rows created.

SQL>

109568 rows created.

SQL>

Commit complete.

SQL> Disconnected … [oracle@MYDBCS ~]$

9 10 11

8

7

6

Database opened. SQL>

Pluggable database altered.

SQL> SQL> Connected. SQL>

Table created.

SQL>

Table created.

…

SQL> SQL>

Table altered.

SQL>

Table altered.

…

SQL> SQL> 2 3 4 5

1. Check the fullness level of the TBSALERT tablespace to see if the warning level has been reached.
   1. Start SQL\*Plus and connect to PDB1 as the SYSTEM user.

$ **sqlplus SYSTEM/*password*@PDB1**

…

SQL>

* 1. Query the size of the TBSALERT tablespace. The results show that the tablespace is 60% full.

SQL> **SELECT sum(bytes) \* 100 / 125829120 FROM dba\_extents WHERE tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

------------------------

60

SQL>

* 1. Query the number of free bytes that are left in the TBSALERT tablespace by executing the $HOME/labs/TBSALERT\_free\_space.sql script . Recall that you created the tablespace with 120MB (125829120 bytes) of space. The query result shows that there are 125829120 bytes free and the tablespace is 39% free.

------------------------------ ---------- ---------- ---------- TBSALERT 49283072 125829120 39.1666667

SQL>

BYTES PCT\_FREE

FREE

TABLESPACE

df.tablespace\_name = 'TBSALERT';

AND

FROM dba\_data\_files df, dba\_free\_space fs

WHERE df.tablespace\_name = fs.tablespace\_name

fs.bytes \*100/ df.bytes PCT\_FREE

df.bytes,

2

3

4

5

SQL> **set echo on**

SQL> **@$HOME/labs/TBSALERT\_free\_space**

SQL> SELECT df.tablespace\_name tablespace, fs.bytes free,

* 1. Wait a few minutes.
  2. Query the DBA\_OUTSTANDING\_ALERTS view to see if there are any new messages. The REASON column is updated with a message stating that the tablespace is 60 percent full. This message is there because the warning level for the tablespace has been reached. If your result is “no rows selected,” wait a little longer and repeat the query.

SQL> **SELECT reason FROM dba\_outstanding\_alerts WHERE object\_name='TBSALERT';**

no rows selected

SQL> **SELECT reason FROM dba\_outstanding\_alerts WHERE object\_name='TBSALERT';**

REASON

----------------------------------------------------------------

Tablespace [TBSALERT@PDB1] is [60 percent] full

SQL>

##### Set a Critical Threshold

In this section, you add more data to the TBSALERT tablespace and check the tablespace fullness threshold again.

1. Execute and commit the following INSERT statements.

SQL> **INSERT INTO hr.employees4 SELECT \* FROM hr.employees4;**

109568 rows created. SQL> **COMMIT;**

Commit complete.

SQL> **INSERT INTO hr.employees5 SELECT \* FROM hr.employees5;**

109568 rows created. SQL> **COMMIT;**

Commit complete.

SQL>

1. Wait a few minutes.
2. Query the fullness of the tablespace. The result shows that the tablespace is 75% full.

SQL> **SELECT sum(bytes) \* 100 / 125829120**

1. **FROM dba\_extents**
2. **WHERE tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

------------------------

75

SQL>

1. Query the outstanding alerts. The REASON column is updated with a message that states the tablespace is 75 percent full. If your result still displays 60, wait a little longer and repeat the query.

SQL> **SELECT reason FROM dba\_outstanding\_alerts WHERE object\_name='TBSALERT';**

REASON

----------------------------------------------------------------

Tablespace [TBSALERT@PDB1] is [75 percent] full SQL>

1. Delete rows from three tables in the HR schema to try to reduce the space used in the tablespace.

SQL> **DELETE hr.employees1;**

219136 rows deleted. SQL> **COMMIT;**

Commit complete.

SQL> **DELETE hr.employees2;**

219136 rows deleted. SQL> **COMMIT;**

Commit complete.

SQL> **DELETE hr.employees3;**

219136 rows deleted SQL> **COMMIT;**

Commit complete.

SQL>

1. Check if there is some reclaimed space after these tables were deleted. The query result indicates that this is not the case. The tablespace is still 75 percent full. Deleting rows frees space in blocks, but it does not return blocks to the tablespace.

SQL> **SELECT sum(bytes) \* 100 / 125829120**

1. **FROM dba\_extents**
2. **WHERE tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

------------------------

75

SQL>

##### Create a Segment Advisor Task

1. Create a Segment Advisor task to get recommendations about the current space situation by executing the **$HOME/labs/seg\_advsr\_task.sql** script.

segments in TBSALERT';

1. task\_id NUMBER;
2. object\_id NUMBER;
3. objectname VARCHAR2(100);
4. objecttype VARCHAR2(100);
5. BEGIN
6. dbms\_advisor.create\_task('Segment Advisor', task\_id,tname,tname\_desc,NULL);
7. dbms\_advisor.create\_object(tname,'TABLESPACE','TBSALERT',' ',' ',NULL,' ', object\_id);

11

dbms\_advisor.set\_task\_parameter(tname,'RECOMMEND\_ALL','TRUE');

12 END; 13 /

PL/SQL procedure successfully completed.

SQL>

VARCHAR2(128) := 'Get shrink advice for

tname\_desc

tname VARCHAR2(128) := 'my\_seg\_task';

2

3

SQL> **set echo on**

SQL> **@$HOME/labs/seg\_advsr\_task.sql**

SQL> DECLARE

1. Execute the task.

PL/SQL procedure successfully completed.

SQL>

**tname VARCHAR2(128) := 'my\_seg\_task'; BEGIN**

**dbms\_advisor.EXECUTE\_TASK(tname); END;**

**/**

2

3

4

5

6

SQL> **DECLARE**

1. Query the DBA\_ADVISOR\_TASKS view for recommendations. The recommendation is to get shrink advice for segments stored in the tablespace.

SQL> **SELECT DESCRIPTION FROM dba\_advisor\_tasks WHERE TASK\_NAME='my\_seg\_task';**

DESCRIPTION

----------------------------------------------------------------

Get shrink advice for segments in TBSALERT

SQL>

1. Execute the $HOME/labs/segments\_to\_shrink.sql script to find out which segments should be shrunk to reclaim space. The result shows that the first three segments should be shrunk.

SQL> **col attr1 format a5** SQL> **col attr2 format a15** SQL> **col message format a55** SQL> **set echo on**

##### SQL> @/home/oracle/labs/segments\_to\_shrink

SQL> SELECT attr1, attr2, message

1. FROM dba\_advisor\_findings f, dba\_advisor\_objects o
2. WHERE f.task\_name = o.task\_name AND f.object\_id = o.object\_id AND f.task\_name = 'my\_seg\_task';

ATTR1 ATTR2 MESSAGE

----- --------------- ------------------------------------------

HR EMPLOYEES3 Perform shrink, estimated savings is 18873242 bytes.

HR EMPLOYEES1 Perform shrink, estimated savings is 18873242 bytes.

HR EMPLOYEES2 Perform shrink, estimated savings is 18873242 bytes.

HR EMPLOYEES4 The free space in the object is less than 10MB.

HR EMPLOYEES5 The free space in the object is less than 10MB.

SQL>

1. Proceed with the SHRINK operation on the HR.EMPLOYEES1, HR.EMPLOYEES2, and

HR.EMPLOYEES3 tables.

SQL> **ALTER TABLE hr.employees1 SHRINK SPACE;**

Table altered.

SQL> **ALTER TABLE hr.employees2 SHRINK SPACE;**

Table altered.

SQL> **ALTER TABLE hr.employees3 SHRINK SPACE;**

Table altered.

SQL>

1. Check if the SHRINK operations reclaimed unused space by running the following query. The result shows that the tablespace did reclaim unused space. It went down to 30% full from 75% full.

SQL> **SELECT sum(bytes) \* 100 / 125829120**

1. **FROM dba\_extents**
2. **WHERE tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

------------------------

30.15625

SQL>

1. Drop the TBSALERT tablespace.

SQL> **DROP TABLESPACE tbsalert INCLUDING CONTENTS AND DATAFILES;**

Tablespace dropped. SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS ~]$

## Practice 13-2: Using Compression

### Overview

In this practice, you will use Advanced Index Compression to reduce the storage for indexes. You use the Compression Advisor, provided by the DBMS\_COMPRESSION package, to get detailed space information about compressing the index with different compression levels.

### Assumptions

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

### Tasks

1. Execute the **$HOME/labs/setup\_index.sh** shell script to create an index with low compression on the HR.TEST table in PDB1.

[oracle@MYDBCS ~]$ **$HOME/labs/setup\_index.sh**

…

SQL> SQL> drop table hr.test

\* ERROR at line 1:

ORA-00942: table or view does not exist

SQL>

Table created.

SQL>

1 row created.

… SQL>

Commit complete.

SQL>

Index created.

SQL> SQL> 2

INDEX\_NAME COMPRESSION

------------------ ------------- I\_TEST DISABLED

SQL> Disconnected from Oracle Database 18c Enterprise Edition

…

[oracle@MYDBCS ~]$

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

$ **sqlplus hr/*password*@PDB1**

…

SQL>

1. Query the compression level of the index created on the **HR.TEST** table. The result indicates that compression is disabled, and therefore, the index is not compressed.

-------------------- ------------- I\_TEST DISABLED

SQL>

COMPRESSION

INDEX\_NAME

SQL> **col index\_name format a20**

SQL> **SELECT index\_name, compression FROM user\_indexes WHERE index\_name = 'I\_TEST';**

1. Query the space used by the index created on the **HR.TEST** table. The result indicates that 1152 blocks are used.

SQL> **SELECT blocks FROM user\_segments WHERE segment\_name='I\_TEST';**

BLOCKS

----------

1152

SQL>

1. Exit SQL\*Plus, but keep the terminal window open.

SQL> **EXIT**

1. View the different compression levels that exist in your Oracle Database version. To do this, use the cat command to review the pre-defined SQL script that creates the DBMS\_COMPRESSION package.

dbms\_compression

DESCRIPTION

Contains package specification for the wrapper

NAME

dbmscomp.sql - DBMS Compression package

Rem Rem Rem Rem

Rem

$ **cat $ORACLE\_HOME/rdbms/admin/dbmscomp.sql**

… Rem

grant execute on dbms\_compression to public

/

show errors;

@?/rdbms/admin/sqlsessend.sql [oracle@MYDBCS ~]$

packages with the advisor framework.

integrate these Rem

Rem

…

package and internal prvt\_compression package. We

Rem

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

$ **sqlplus hr/*password*@PDB1**

…

SQL>

1. Use the Compression Advisor to get recommendations about the space you would save by compressing the index with the COMP\_INDEX\_ADVANCED\_LOW compression level by executing the $HOME/labs/Compression\_index\_low.sql script. The result indicates that the space used by the index would be reduced down to 809 blocks. The Advanced Low Compression ratio equals 1.

SQL> **set echo on**

SQL> **@$HOME/labs/Compression\_index\_low.sql**

SQL> set serveroutput on SQL> DECLARE

blkcnt\_cmp pls\_integer; blkcnt\_uncmp pls\_integer; row\_cmp pls\_integer; row\_uncmp pls\_integer; cmp\_ratio pls\_integer; comptype\_str varchar2(100); BEGIN

DBMS\_COMPRESSION.GET\_COMPRESSION\_RATIO

(

scratchtbsname => 'USERS', ownname => 'HR',

objname => 'I\_TEST', subobjname => NULL,

comptype => dbms\_compression.COMP\_INDEX\_ADVANCED\_LOW, blkcnt\_cmp => blkcnt\_cmp,

blkcnt\_uncmp => blkcnt\_uncmp,

row\_cmp => row\_cmp,

row\_uncmp => row\_uncmp, cmp\_ratio => cmp\_ratio, comptype\_str => comptype\_str,

subset\_numrows => dbms\_compression.COMP\_RATIO\_MINROWS, objtype => dbms\_compression.OBJTYPE\_INDEX

);

DBMS\_OUTPUT.PUT\_LINE('Block used by compressed index = ' || blkcnt\_cmp);

DBMS\_OUTPUT.PUT\_LINE('Block used by uncompressed index = ' || blkcnt\_uncmp);

DBMS\_OUTPUT.PUT\_LINE('Compression type = ' || comptype\_str); DBMS\_OUTPUT.PUT\_LINE('Compression ratio org = '||cmp\_ratio); END;

/

Block used by compressed index = 809 Block used by uncompressed index = 1029 Compression type = "Compress Advanced Low" Compression ratio org = 1

PL/SQL procedure successfully completed.

SQL>

1. Use the Compression Advisor again to get recommendations about the space you would save by compressing the index with the COMP\_INDEX\_ADVANCED\_HIGH compression level by executing the $HOME/labs/Compression\_index\_high.sql script. The result indicates that the space used by the index would be reduced down to 130 blocks. The Advanced High Compression ratio is equal to 8.

SQL> **@$HOME/labs/Compression\_index\_high.sql**

SQL> set serveroutput on SQL> DECLARE

blkcnt\_cmp pls\_integer;

blkcnt\_uncmp pls\_integer; row\_cmp pls\_integer; row\_uncmp pls\_integer; cmp\_ratio pls\_integer; comptype\_str varchar2(100); BEGIN

DBMS\_COMPRESSION.GET\_COMPRESSION\_RATIO (

scratchtbsname => 'USERS', ownname => 'HR',

objname => 'I\_TEST',

subobjname => NULL,

comptype => dbms\_compression.COMP\_INDEX\_ADVANCED\_HIGH, blkcnt\_cmp => blkcnt\_cmp,

blkcnt\_uncmp => blkcnt\_uncmp,

row\_cmp => row\_cmp, row\_uncmp => row\_uncmp, cmp\_ratio => cmp\_ratio, comptype\_str => comptype\_str,

subset\_numrows => dbms\_compression.COMP\_RATIO\_MINROWS,

objtype => dbms\_compression.OBJTYPE\_INDEX

);

DBMS\_OUTPUT.PUT\_LINE('Block used by compressed index = ' || blkcnt\_cmp);

DBMS\_OUTPUT.PUT\_LINE('Block used by uncompressed index = ' || blkcnt\_uncmp);

DBMS\_OUTPUT.PUT\_LINE('Compression type = ' || comptype\_str); DBMS\_OUTPUT.PUT\_LINE('Compression ratio org = '||cmp\_ratio); END;

/

Block used by compressed index = 130 Block used by uncompressed index = 1029

Compression type = "Compress Advanced High" Compression ratio org = 8

PL/SQL procedure successfully completed.

SQL>

1. Question: Based on the previous steps, which compression ratio is the best—the COMP\_INDEX\_ADVANCED\_LOW or COMP\_INDEX\_ADVANCED\_HIGH compression level? Answer: The Advanced High Compression ratio (8) is much better than the Advanced Low

Compression ratio (1). Therefore, you would be inclined to rebuild the index with Advanced High Compression.

1. Rebuild the index with Advanced High Compression.

SQL> **ALTER INDEX hr.i\_test REBUILD COMPRESS ADVANCED HIGH;**

Index altered.

SQL>

1. Query the compression level of the index created on the HR.TEST table. The result shows that the compression level is ADVANCED HIGH.

SQL> **col index\_name format a20**

-------------------- ------------- I\_TEST ADVANCED HIGH

SQL>

COMPRESSION

INDEX\_NAME

SQL> **SELECT index\_name, compression FROM user\_indexes WHERE index\_name = 'I\_TEST';**

1. Query the space used by the index created on the **HR.TEST** table. The space is now 256 blocks.

SQL> **SELECT blocks FROM user\_segments WHERE segment\_name='I\_TEST';**

BLOCKS

----------

256

SQL>

1. Question: Is it possible to revert back to the initial compression level? Answer: Yes.
2. Revert back to the initial compression level.

SQL> **ALTER INDEX hr.i\_test REBUILD NOCOMPRESS;**

Index altered.

SQL>

1. Query the space used by the index created on the HR.TEST table. The space used is 1152 blocks again.

SQL> **SELECT blocks FROM user\_segments WHERE segment\_name='I\_TEST';**

BLOCKS

----------

1152

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

## Practice 13-3: Enabling the Resumable Space Allocation Feature

### Overview

In this practice, you enable the Resumable Space Allocation feature to avoid situations where a tablespace runs out of space and causes operations to fail; for example, rows cannot be loaded into a table. You will work in two terminal windows (window 1 and window 2).

With the Resumable Space Allocation feature:

* + Some operations are resumable, but not all. These operations are called resumable statements. INSERT, INSERT INTO SELECT, UPDATE, and DELETE statements are candidates.
  + Some errors are correctable, but not all; for example: out of space condition (ORA- 01653, ORA-01654), maximum extents reached condition (ORA-01631, ORA-01632), space quota exceeded condition (ORA-01536).

##### Tip

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

To set a title for a terminal window:

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

### Assumptions

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

### Tasks

##### Window 1: Enable the Resumable Space Allocation Feature

1. In your open terminal window, start SQL\*Plus and connect to PDB1 as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus SYSTEM/*password*@PDB1**

…

SQL>

1. Grant the PDBADMIN user the DBA role. If you completed Practice 9-3 Granting the DBA

Role to PDBADMIN, you can skip this step.

SQL> **GRANT DBA TO PDBADMIN;**

Grant succeeded.

SQL>

1. Connect to PDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the password value.

SQL> **CONNECT PDBADMIN/*password*@PDB1**

Connected. SQL>

1. Execute the $HOME/labs/CreateINVENTORYTablespace.sql script to create an unpopulated tablespace named INVENTORY.

SQL> **@$HOME/labs/CreateINVENTORYTablespace.sql**

Tablespace created. SQL>

1. Execute the $HOME/labs/CreateTable\_X.sql script to create and populate a table named X in the INVENTORY tablespace. As the script runs, notice that rows are being inserted into the table. Part way through the script, you get an error telling you that there is not enough space in the INVENTORY tablespace to insert the remaining rows.

SQL> **@$HOME/labs/CreateTable\_X.sql**

PL/SQL procedure successfully completed. SQL> CREATE TABLE x

2 (a CHAR(1000)

3 ) TABLESPACE inventory; Table created.

SQL> INSERT INTO x

2 VALUES ('a');

1 row created.

…

SQL> INSERT INTO x

2 SELECT \* FROM x ;

1024 rows created. SQL> INSERT INTO x

2 SELECT \* FROM x ; INSERT INTO x

\*

ERROR at line 1:

ORA-01653: unable to extend table PDBADMIN.X by 128 in tablespace INVENTORY

SQL> COMMIT;

Commit complete.

SQL> quit

…

[oracle@MYDBCS ~]$

1. Imagine that the operation in the previous step had lasted 5 hours and that the load had nearly reached its end and other operations were depending on its success.
   1. Question: Are the rows that were inserted into the table lost or definitely inserted? Answer: 2048 rows were inserted.
   2. Question: How could this situation be avoided when you do not know how much space is required for a table to load all its rows?

Answer: In the case of heavy load operations, you can use a corrective action rather than a reactive action after an error is raised. For example, you can use the Resumable Space Allocation feature.

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

$ **sqlplus PDBADMIN/*password*@PDB1**

…

SQL>

1. Enable resumable mode.

SQL> **ALTER SESSION ENABLE RESUMABLE;**

Session altered.

SQL>

1. Re-execute the CreateTable\_X script. The script is suspended.

SQL> **@$HOME/labs/CreateTable\_X.sql**

PL/SQL procedure successfully completed. SQL> CREATE TABLE x

2 (a CHAR(1000)

3 ) TABLESPACE inventory;

Table created.

SQL> INSERT INTO x

2 VALUES ('a');

1 row created.

…

SQL> INSERT INTO x

2 SELECT \* FROM x ;

1024 rows created. SQL> INSERT INTO x

2 SELECT \* FROM x ;

1. Question: Why is the script suspended?

Answer: Enabling the resumable mode for your session suspends the failing statement during 7200 seconds (2 hours), by default.

1. Question: Is there any warning message to tell you the load is suspended? Answer: No. If the script does not execute any further, check the alert log file or the

DBA\_RESUMABLE view. An operation-suspended alert is issued on the object that needs

allocation of resource for the operation to complete.

##### Window 2: Resolve a Suspended Script

1. Open another terminal window. This will be referred to as Window 2. Connect to the compute node as the oracle user.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Start SQL\*Plus and connect to PDB1 as SYSTEM.

$ **sqlplus SYSTEM/*password*@PDB1**

…

SQL>

1. Query the DBA\_RESUMABLE view for information about the suspended script. The DBA\_RESUMABLE view lists all resumable statements executed in the system. Your times and session information will be different from those shown below.

SQL> **set pages 100**

SQL> **SELECT status, name, sql\_text, error\_msg FROM dba\_resumable;**

STATUS

--------- NAME

---------------------------------------------------------------- SQL\_TEXT

---------------------------------------------------------------- ERROR\_MSG

---------------------------------------------------------------- SUSPENDED

User PDBADMIN(109), Session 278, Instance 1 INSERT INTO x SELECT \* FROM x

ORA-01653: unable to extend table PDBADMIN.X by 128 in tablespace INVENTORY

SQL>

1. Exit SQL\*Plus, but keep the terminal window open.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Check the alert log file for information about the suspended script. The log states that the suspension occurred because the table could not be extended.

[oracle@MYDBCS ~]$ **tail -30**

**/u01/app/oracle/diag/rdbms/orcl/ORCL/trace/alert\_ORCL.log**

…

PDB1(4):ORA-1653: unable to extend table PDBADMIN.X by 128 in tablespace INVENTORY [PDB1]

2018-03-30T18:22:04.168473+00:00

PDB1(4):statement in resumable session 'User PDBADMIN(109), Session 278, Instance 1' was suspended due to

PDB1(4): ORA-01653: unable to extend table PDBADMIN.X by 128 in tablespace INVENTORY

2018-03-30T18:27:22.901017+00:00

Control autobackup written to DISK device

handle '/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_03\_30/o 1\_mf\_s\_972152836\_fcx0d7gn\_.bkp'

[oracle@MYDBCS ~]$

1. Proceed with the appropriate corrective action. Because the INVENTORY tablespace is not autoextensible, you can configure it as autoextensible with a size limit.
   1. Start SQL\*Plus and connect to PDB1 as the SYSTEM user.

$ **sqlplus SYSTEM/*password*@PDB1**

… SQL>

* 1. Query the DBA\_DATA\_FILES view to verify whether the INVENTORY tablespace is autoextensible. The result shows that the tablespace is not.

SQL> **COL file\_name FORMAT A60**

SQL> **SELECT file\_name, autoextensible FROM dba\_data\_files WHERE tablespace\_name='INVENTORY';**

FILE\_NAME AUT

------------------------------------------------------------ ---

/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF NO

SQL>

* 1. Enable autoextend for the INVENTORY01.DBF data file.

SQL> **ALTER DATABASE DATAFILE**

**'/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF' AUTOEXTEND ON MAXSIZE 10M;**

Database altered. SQL>

* 1. Query the DBA\_DATA\_FILES view again to verify whether the INVENTORY tablespace is autoextensible. The result shows that it is.

SQL> **COL file\_name FORMAT A60**

SQL> **SELECT file\_name, autoextensible FROM dba\_data\_files WHERE tablespace\_name='INVENTORY';**

FILE\_NAME AUT

------------------------------------------------------------ ---

/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF YES

SQL>

##### Window 1: Check the Suspended Session

1. Return to Window 1. Notice that the session is no longer suspended. The results show that 2048 rows were created, and the transaction was committed. After the resource had been allocated, the operation completed, and the operation-suspended alert cleared.

SQL> INSERT INTO x

2 SELECT \* FROM x ; 2048 rows created. SQL> COMMIT;

Commit complete.

SQL> quit

…

[oracle@MYDBCS ~]$

1. Close the terminal window.

##### Window 2: Verify that there are no Suspended Sessions

1. Return to Window 2. Verify that there are no suspended sessions in the system by querying the DBA\_RESUMABLE view again.

SQL> **SELECT status, name, sql\_text, error\_msg FROM dba\_resumable;**

no rows selected SQL>

1. Exit from SQL\*Plus and close the terminal window.

SQL> **EXIT**

**Practices for Lesson 14:**

**Managing Undo Data**

## Practices for Lesson 14: Overview

### Overview

In these practices, you will view undo activity and configure the database to support twelve-hour retention for flashback operations.

## Practice 14-1: Managing Undo Data

### Overview

In this practice, you first view your system activity regarding the undo feature, and then you configure the ORCL database to support twelve-hour retention for flashback operations.

Enterprise Manager Database Express (EM Express) enables you to change the undo tablespace and perform analysis.

##### Tip

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

To set a title for a terminal window:

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

### Assumptions

You are logged in as the oracle user.

### Tasks

##### Window 1: Start a Workload in the Database

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Execute the $HOME/labs/UNDO\_setup.sh shell script to set the UNDO tablespace in the root container to a fixed size.

[oracle@MYDBCS ~]$ **$HOME/labs/UNDO\_setup.sh**

Database altered. [oracle@MYDBCS ~]$

1. Execute the $HOME/labs/UNDO\_setup\_tuning.sh shell script. This script creates a user named OE in PDB1 and grants that user the DBA role. It creates a tablespace named TBS\_APP in PDB1 and creates several tables in the TBS\_APP tablespace.

[oracle@MYDBCS ~]$ **$HOME/labs/UNDO\_setup\_tuning.sh**

…

Commit complete. [oracle@MYDBCS ~]$

##### Window 2:

1. Open a new terminal window and connect to the compute node as the oracle user.
2. Source the oraenv script.
3. Start SQL\*Plus and connect to PDB1 as the OE user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus oe/password@PDB1**

…

SQL>

1. Perform two UPDATE commands in PDB1. The first update may take a minute or two.

SQL> **UPDATE oe.lineorder set LO\_QUANTITY = LO\_QUANTITY\*10;**

3297032 rows updated.

SQL> **UPDATE oe.product\_information set min\_price = min\_price\*10;**

287 rows updated. SQL>

##### Window 3: Start a Workload

1. Open another terminal window and connect to the compute node as the oracle user.
2. Source the oraenv script.
3. Execute the $HOME/labs/UNDO\_loop.sh shell script to start a workload in the database.

[oracle@MYDBCS ~]$ **$HOME/labs/UNDO\_loop.sh**

##### Window 4: Start a Workload

1. Open another terminal window.
2. Execute the $HOME/labs/UNDO\_loop.sh shell script to start a workload in the database.

[oracle@MYDBCS ~]$ **$HOME/labs/UNDO\_loop.sh**

##### Window 5: Start a Workload

1. Open another terminal window.
2. Execute the $HOME/labs/UNDO\_loop.sh shell script to start a workload in the database.

[oracle@MYDBCS ~]$ **$HOME/labs/UNDO\_loop.sh**

##### Use Enterprise Manager Database Express to Analyze Undo Data

1. Open a new terminal window and create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

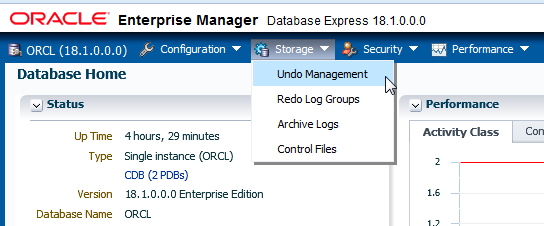
[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

[oracle@MYDBCS ~]$

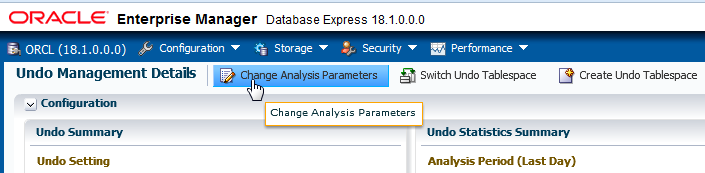
1. Open a browser and launch Enterprise Manager Database Express by entering the following URL: **https://localhost:5500/em**
2. Log in as the **SYSTEM** user. Refer to *Course Practice Environment: Security Credentials*

for the password value.

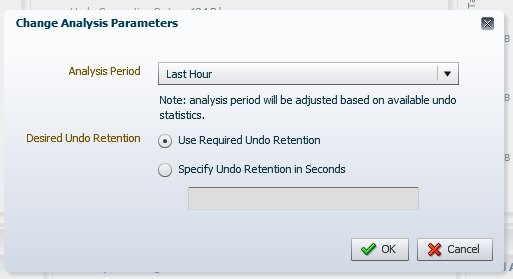
1. Select **Storage** and then **Undo Management**.



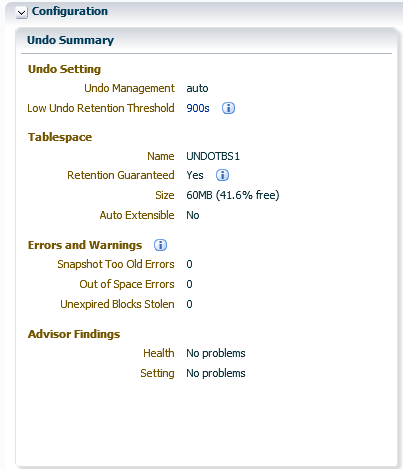
1. Click **Change Analysis Parameters**.



1. In the Analysis Period drop-down list, select **Last Hour**. Leave **Use Required Undo Retention** selected and click **OK**.



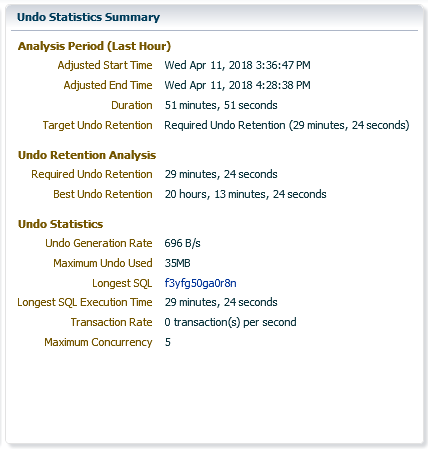
1. In the Confirmation dialog box, click **OK**.
2. In the Undo Summary pane, review the undo configuration. Your tablespace size may be different.



1. Question: How many errors did this system encounter?

Answer: None. Even if the undo tablespace is currently set to a fixed size, there are no failed transactions.

1. In the Undo Statistics Summary pane, review the information. Your values will be different from those shown below.



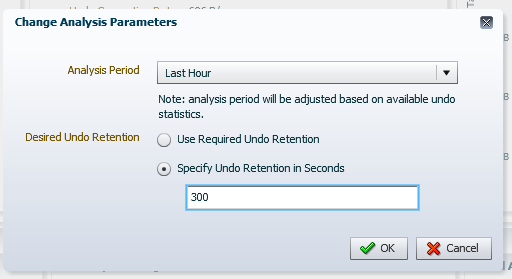
1. Question: What is the duration of the longest running query?

Answer: In this example, the longest running query is 29 minutes. Your value will be different.

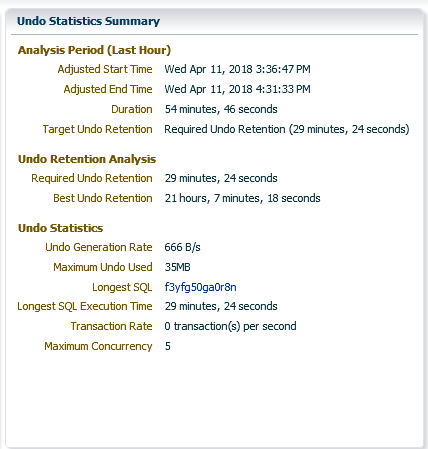
1. Question: What would be the best undo retention for the past hour statistics collected and the current configuration?

Answer: In this example, the best undo retention is 20 hours and 13 minutes. Your value will be different.

1. Restart an analysis, keeping the past hour as the period of analysis and defining the Undo Retention to 300 seconds (5 minutes).
   1. Click **Change Analysis Parameters**.
   2. In the Change Analysis Parameters dialog box, select Last Hour from the Analysis Period drop-down list, select the **Specify Undo Retention in Seconds** option, and enter **300**. Click **OK**.



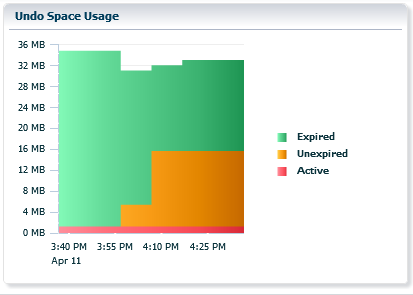
* 1. In the Confirmation dialog box, click **OK**.
  2. In the Undo Statistics Summary pane, review the information (particularly the Undo Retention Analysis section).



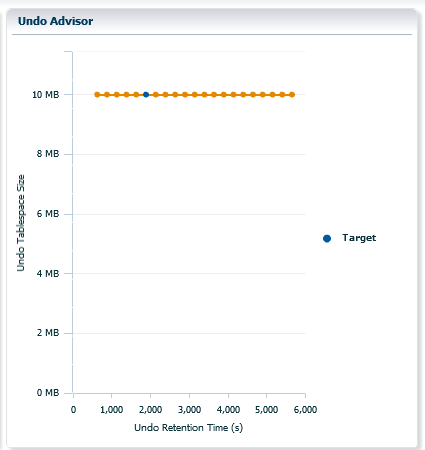
* 1. Is the best undo retention time the same as before?

Answer: No, not in this example. With the undo retention configured to 5 minutes, the best undo retention time you may get is 21 hours.

* 1. At the bottom of the page, observe the Undo Space Usage during the past hour. There are very few active transactions. But there is a fairly large amount of committed undo information (Unexpired) still needed to meet the undo retention interval. Your graphs will look different from those shown below.



* 1. On the right side of the page, the Undo Advisor shows its recommendation. In this example, the undo tablespace size should be at least 10MB. The blue point shows the current undo retention value and the undo tablespace size.



* 1. When you have finished analyzing the undo data, log out of EM Express and close the browser window.

##### Window 1: Close the Window

1. Close the connection to the compute node.
2. Close the terminal window.

##### Window 2: Roll Back the Transaction

1. Roll back the UPDATE transaction you started earlier. The rollback may take a minute or two to complete.

SQL> **ROLLBACK;**

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

SQL> **EXIT**

##### Windows 3, 4, and 5: Close the Window

1. If the workload is still running, cancel the script by entering Ctrl + C.
2. Close the connection to the compute node.
3. Close the terminal window.

**Practices for Lesson 15:**

**Moving Data**

**Practices for Lesson 15: Overview**

### Overview

In these practices, you will move data from one PDB to another PDB.

**Practice 15-1: Moving Data from One PDB to Another PDB**

### Overview

In this practice, imagine that you configured PDB2 with different optimizer parameter values, and you want to test the performance of requests on OE tables in PDB2 to compare it with the performance of the same queries in PDB1. Through trial and error, you export all objects from the OE schema from PDB1 and import them into PDB2 under a new schema named OETEST for testing purposes.

### Assumptions

You are logged in as the oracle user.

### Tasks

##### Export the OE Schema from PDB1 by Using Data Pump Export

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Create a new directory named PDB2.

[oracle@MYDBCS ~]$ **mkdir /u02/app/oracle/oradata/ORCL/PDB2**

[oracle@MYDBCS ~]$

1. Execute the $HOME/labs/DP\_setup.sh shell script to create tables in PDB1 and PDB2.

[oracle@MYDBCS ~]$ **$HOME/labs/DP\_setup.sh**

Tablespace dropped. Tablespace created.

…

1 row created.

Commit complete.

[oracle@MYDBCS ~]$

1. Launch Data Pump export under a connection as OE in PDB1 to export all objects belonging to OE. Refer to *Course Practice Environment: Security Credentials* for the password value. Use the DUMPFILE parameter to specify the location and name of the dump file resulting from the export operation.

You will get an error during this operation stating that the file name cannot contain a path specification.

[oracle@MYDBCS ~]$ **expdp oe/*password*@PDB1 SCHEMAS=oe DUMPFILE=/u01/app/oracle/admin/ORCL/dpdump/expoe.dmp**

Export: Release 18.0.0.0.0 - Production on Fri Apr 13 11:48:08 2018

Version 18.1.0.0.0

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

Connected to: Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

ORA-39001: invalid argument value

ORA-39000: bad dump file specification

ORA-39088: file name cannot contain a path specification

[oracle@MYDBCS ~]$

1. Question: What does the error message lead you to do?

Answer: Create a logical directory in PDB1. Directory objects are required when you specify file locations for Data Pump because it accesses files on the server rather than on the client. Directory objects are logical structures that represent a physical directory on the server’s file system. They contain the location of a specific operating system directory.

Directory objects are owned by the SYS user. Directory names are unique across the database because all the directories are located in a single name space.

1. Start SQL\*Plus and connect to PDB1 as the SYS user with the SYSDBA privilege. Refer to

*Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus SYS/*password*@PDB1 AS SYSDBA**

… SQL>

1. Create a logical directory named DP\_FOR\_OE in PDB1.

SQL> **CREATE DIRECTORY dp\_for\_oe AS '/u01/app/oracle/admin/ORCL/dpdump';**

Directory created.

SQL>

1. Grant the OE user READ WRITE privileges on the DP\_FOR\_OE directory.

SQL> **GRANT read, write ON DIRECTORY dp\_for\_oe TO oe;**

Grant succeeded.

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Retry the Data Pump export. Refer to *Course Practice Environment: Security Credentials*

for the password value.

[oracle@MYDBCS ~]$ **expdp oe/*password*@PDB1 SCHEMAS=oe DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp**

Export: Release 18.0.0.0.0 - Production on Fri Apr 13 11:51:16 2018

Version 18.1.0.0.0

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

Connected to: Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

Starting "OE"."SYS\_EXPORT\_SCHEMA\_01": oe/\*\*\*\*\*\*\*\*@PDB1 SCHEMAS=oe DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp Processing object type SCHEMA\_EXPORT/TABLE/TABLE\_DATA

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

Processing object type SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

Processing object type SCHEMA\_EXPORT/STATISTICS/MARKER Processing object type SCHEMA\_EXPORT/USER

Processing object type SCHEMA\_EXPORT/SYSTEM\_GRANT Processing object type SCHEMA\_EXPORT/ROLE\_GRANT Processing object type SCHEMA\_EXPORT/DEFAULT\_ROLE

Processing object type SCHEMA\_EXPORT/PRE\_SCHEMA/PROCACT\_SCHEMA Processing object type SCHEMA\_EXPORT/SEQUENCE/SEQUENCE Processing object type SCHEMA\_EXPORT/TABLE/TABLE

Processing object type SCHEMA\_EXPORT/TABLE/COMMENT Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX

Processing object type SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

KB 105 rows

. . exported "OE"."ORDER\_ITEMS" 21.01

KB 665 rows

ORA-39173: Encrypted data has been stored unencrypted in dump file set.

Master table "OE"."SYS\_EXPORT\_SCHEMA\_01" successfully loaded/unloaded

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Dump file set for OE.SYS\_EXPORT\_SCHEMA\_01 is:

/u01/app/oracle/admin/ORCL/dpdump/expoe.dmp

Job "OE"."SYS\_EXPORT\_SCHEMA\_01" successfully completed at Fri Apr 13 11:52:20 2018 elapsed 0 00:01:03

[oracle@MYDBCS ~]$

12.73

. . exported "OE"."ORDERS"

1. Use the oerr utility to determine whether the ORA-39173 error is of concern.

// secure data may be readable from within the dump file set.

[oracle@MYDBCS ~]$

This is only a

involved data that was encrypted in the database.

// \*Action: No specific user action is required. warning that

No encryption password was specified for an export

// \*Cause: job that

//

[oracle@MYDBCS ~]$ **oerr ora 39173**

39173, 00000, "Encrypted data has been stored unencrypted in dump file set."

1. Question: How can you verify that objects other than tables, such as constraints, indexes, and sequences, were exported?

Answer: Generate a SQL script from the dump file by performing an import and specifying the SQLFILE parameter.

1. Use Data Pump Import to generate a SQL script named oe\_SQL.sql from the dump file. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **impdp oe/*password*@PDB1 SCHEMAS=oe DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp SQLFILE=oe\_SQL**

Import: Release 18.0.0.0.0 - Production on Fri Apr 13 12:01:52 2018

Version 18.1.0.0.0

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

Connected to: Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

Master table "OE"."SYS\_SQL\_FILE\_SCHEMA\_01" successfully loaded/unloaded

Starting "OE"."SYS\_SQL\_FILE\_SCHEMA\_01": oe/\*\*\*\*\*\*\*\*@PDB1 SCHEMAS=oe DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp SQLFILE=oe\_SQL Processing object type SCHEMA\_EXPORT/USER

Processing object type SCHEMA\_EXPORT/SYSTEM\_GRANT Processing object type SCHEMA\_EXPORT/ROLE\_GRANT Processing object type SCHEMA\_EXPORT/DEFAULT\_ROLE

Processing object type SCHEMA\_EXPORT/PRE\_SCHEMA/PROCACT\_SCHEMA Processing object type SCHEMA\_EXPORT/SEQUENCE/SEQUENCE Processing object type SCHEMA\_EXPORT/TABLE/TABLE

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX Processing object type SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

Processing object type SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

Processing object type SCHEMA\_EXPORT/STATISTICS/MARKER

Job "OE"."SYS\_SQL\_FILE\_SCHEMA\_01" successfully completed at Fri Apr 13 12:02:00 2018 elapsed 0 00:00:06

[oracle@MYDBCS ~]$

1. Review the oe\_SQL.sql script. When you execute the import, all DDL statements are read from the dump file.

$ more /u01/app/oracle/admin/ORCL/dpdump/oe\_SQL.sql

-- CONNECT OE

ALTER SESSION SET EVENTS '10150 TRACE NAME CONTEXT FOREVER, LEVEL 1';

ALTER SESSION SET EVENTS '10904 TRACE NAME CONTEXT FOREVER, LEVEL 1';

ALTER SESSION SET EVENTS '25475 TRACE NAME CONTEXT FOREVER, LEVEL 1';

ALTER SESSION SET EVENTS '10407 TRACE NAME CONTEXT FOREVER, LEVEL 1';

ALTER SESSION SET EVENTS '10851 TRACE NAME CONTEXT FOREVER, LEVEL 1';

ALTER SESSION SET EVENTS '22830 TRACE NAME CONTEXT FOREVER, LEVEL 192 ';

-- new object type path: SCHEMA\_EXPORT/USER

CREATE USER "OE" IDENTIFIED BY VALUES 'S:56EAFD0BC7CC6C0EC651879A424D07AEF64BF9

5D81E1E5EE765E520803F9;T:A4E7A0B274BFCEEA000FD2466A39D3D454D1280

7FEE3D9E20C5201F

2E220FDAA1A594568A4D591602C9DCA0AB7D194D3002C221DD6D1254A85C86DB

060402A65A078357

81DF75F47E03F9C5466F474AB'

DEFAULT TABLESPACE "TBS\_APP" TEMPORARY TABLESPACE "TEMP";

…

-- new object type path: SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

ALTER TABLE "OE"."ORDERS" ADD PRIMARY KEY ("ORDER\_ID") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS

2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "TBS\_APP" ENABLE;

-- new object type path: SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

-- new object type path: SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

-- new object type path: SCHEMA\_EXPORT/STATISTICS/MARKER [oracle@MYDBCS ~]$

##### Import the OE Schema into PDB2 by Using Data Pump Import

1. Start SQL\*Plus and connect to PDB2 as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus SYSTEM/*password*@PDB2**

…

SQL>

1. In case the OETEST schema already exists in PDB2, execute the DROP USER command to drop the OETEST user.

SQL> **DROP USER oetest CASCADE;**

DROP USER oetest CASCADE

\* ERROR at line 1:

ORA-01918: user 'OETEST' does not exist

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Use Data Pump to import the OE schema. Refer to *Course Practice Environment: Security Credentials* for the password value. Use the REMAP\_SCHEMA parameter to import the entire OE schema into a new OETEST schema in PDB2.

You will get an error message stating that the directory name for DP\_FOR\_OE is invalid.

[oracle@MYDBCS ~]$ **impdp SYSTEM/*password*@PDB2 REMAP\_SCHEMA=oe:oetest DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp**

Import: Release 18.0.0.0.0 - Production on Fri Apr 13 12:07:14 2018

Version 18.1.0.0.0

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

Connected to: Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

ORA-39002: invalid operation

ORA-39070: Unable to open the log file.

ORA-39087: directory name DP\_FOR\_OE is invalid

[oracle@MYDBCS ~]$

1. Why did you receive an error message that the directory DP\_FOR\_OE does not exist when you created that directory in a previous step?

Answer: You created the directory in PDB1, not in PDB2.

1. Connect to PDB2 as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus SYSTEM/*password*@PDB2**

…

SQL>

1. Create the DP\_FOR\_OE directory in PDB2.

SQL> **CREATE DIRECTORY dp\_for\_oe AS '/u01/app/oracle/admin/ORCL/dpdump';**

Directory created. SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Retry the import operation. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **impdp SYSTEM/*password*@PDB2 REMAP\_SCHEMA=oe:oetest DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp**

Import: Release 18.0.0.0.0 - Production on Fri Apr 13 12:10:10 2018

Version 18.1.0.0.0

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

Connected to: Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

Master table "SYSTEM"."SYS\_IMPORT\_FULL\_01" successfully loaded/unloaded

Starting "SYSTEM"."SYS\_IMPORT\_FULL\_01": SYSTEM/\*\*\*\*\*\*\*\*@PDB2 REMAP\_SCHEMA=oe:oetest DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp

Processing object type SCHEMA\_EXPORT/USER Processing object type SCHEMA\_EXPORT/SYSTEM\_GRANT Processing object type SCHEMA\_EXPORT/ROLE\_GRANT Processing object type SCHEMA\_EXPORT/DEFAULT\_ROLE

Processing object type SCHEMA\_EXPORT/PRE\_SCHEMA/PROCACT\_SCHEMA Processing object type SCHEMA\_EXPORT/SEQUENCE/SEQUENCE Processing object type SCHEMA\_EXPORT/TABLE/TABLE

ORA-39083: Object type TABLE:"OETEST"."ORDER\_ITEMS" failed to

create with error:

ORA-00959: tablespace 'TBS\_APP2' does not exist

Failing sql is:

CREATE TABLE "OETEST"."ORDER\_ITEMS" ("ORDER\_ID" NUMBER(12,0), "LINE\_ITEM\_ID" NUMBER(3,0), "PRODUCT\_ID" NUMBER(6,0), "UNIT\_PRICE" NUMBER(8,2), "QUANTITY" NUMBER(8,0)) SEGMENT CREATION IMMEDIATE PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS

1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT) TABLESPACE "TBS\_APP2"

Processing object type SCHEMA\_EXPORT/TABLE/TABLE\_DATA

. . imported "OETEST"."ORDERS" 12.73

KB 105 rows

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX

ORA-39112: Dependent object type INDEX:"OETEST"."I\_ORDER\_ITEMS" skipped, base object type TABLE:"OETEST"."ORDER\_ITEMS" creation failed

Processing object type SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

Processing object type SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

Processing object type SCHEMA\_EXPORT/STATISTICS/MARKER

Job "SYSTEM"."SYS\_IMPORT\_FULL\_01" completed with 2 error(s) at Fri Apr 13 12:10:39 2018 elapsed 0 00:00:28

[oracle@MYDBCS ~]$

1. Question: Did the import complete successfully?

Answer: Not completely. Data Pump imported only the objects that it could process without any error.

1. Question: Which objects were not imported?

Answer: Data Pump could not import the ORDER\_ITEMS table because this table requires the TBS\_APP2 tablespace, which does not exist in PDB2. The dependent objects of this table, such as an index could not be imported.

1. Create the TBS\_APP2 tablespace in PDB2.
   1. Start SQL\*Plus and connect to PDB2 as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus SYSTEM/*password*@PDB2**

…

SQL>

* 1. Issue the CREATE TABLESPACE command to create the TBS\_APP2 tablespace in

PDB2.

SQL> **CREATE TABLESPACE tbs\_app2 DATAFILE**

**'/u02/app/oracle/oradata/ORCL/PDB2/tbs\_app02.dbf' SIZE 100m;**

Tablespace created. SQL>

* 1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Retry the import operation. Refer to *Course Practice Environment: Security Credentials* for the password value.

##### $ impdp SYSTEM/*password*@PDB2 REMAP\_SCHEMA=oe:oetest DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp

Import: Release 18.0.0.0.0 - Production on Fri Apr 13 12:14:23 2018

Version 18.1.0.0.0

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

Connected to: Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

Master table "SYSTEM"."SYS\_IMPORT\_FULL\_01" successfully loaded/unloaded

Starting "SYSTEM"."SYS\_IMPORT\_FULL\_01": SYSTEM/\*\*\*\*\*\*\*\*@PDB2 REMAP\_SCHEMA=oe:oetest DIRECTORY=dp\_for\_oe DUMPFILE=expoe.dmp

Processing object type SCHEMA\_EXPORT/USER

ORA-31684: Object type USER:"OETEST" already exists

Processing object type SCHEMA\_EXPORT/SYSTEM\_GRANT Processing object type SCHEMA\_EXPORT/ROLE\_GRANT Processing object type SCHEMA\_EXPORT/DEFAULT\_ROLE

Processing object type SCHEMA\_EXPORT/PRE\_SCHEMA/PROCACT\_SCHEMA Processing object type SCHEMA\_EXPORT/SEQUENCE/SEQUENCE

ORA-31684: Object type SEQUENCE:"OETEST"."ORDERS\_SEQ" already

exists

Processing object type SCHEMA\_EXPORT/TABLE/TABLE

ORA-39151: Table "OETEST"."ORDERS" exists. All dependent metadata and data will be skipped due to table\_exists\_action of skip

Processing object type SCHEMA\_EXPORT/TABLE/TABLE\_DATA

. . imported "OETEST"."ORDER\_ITEMS" 21.01

KB 665 rows

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX Processing object type SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

Processing object type SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

Processing object type SCHEMA\_EXPORT/STATISTICS/MARKER

Job "SYSTEM"."SYS\_IMPORT\_FULL\_01" completed with 3 error(s) at Fri Apr 13 12:14:47 2018 elapsed 0 00:00:22

[oracle@MYDBCS ~]$

1. Question: Are the errors true errors?

Answer: The errors are normal errors stating that objects exist. They were created during the previous import operation.

##### Verify the OETEST Schema in PDB2

Verify that the new OETEST schema exists in PDB2.

1. Start SQL\*Plus and connect to PDB2 as the OETEST user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus oetest/*password*@PDB2**

…

SQL>

1. View the list of tables to which the OETEST user has access.

SQL> **SELECT table\_name FROM user\_tables;**

TABLE\_NAME

---------------------------------------------------------------- ORDERS

ORDER\_ITEMS

SQL>

1. Query the number of rows in the ORDER\_ITEMS table. The results show that there are 665 rows.

SQL> **SELECT count(\*) FROM order\_items;**

COUNT(\*)

----------

665

SQL>

1. List the indexes to which the OETEST user has access.

SQL> **SELECT index\_name FROM user\_indexes;**

INDEX\_NAME

---------------------------------------------------------------- SYS\_C0011373

I\_ORDER\_ITEMS

SQL>

1. List the sequences to which the OETEST user has access.

SQL> **SELECT sequence\_name FROM user\_sequences;**

SEQUENCE\_NAME

---------------------------------------------------------------- ORDERS\_SEQ

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Question: How could you have imported the OE schema from PDB1 to PDB2 in one single operation?

Answer: The data could be imported from PDB1 by using a valid database link and written directly back to the connected PDB2. The Data Pump import operation uses the NETWORK\_LINK parameter to define the database link used to access the database from which to import the data.

##### Import the OE Schema into PDB2 via a Database Link

1. Start SQL\*Plus and connect to PDB2 as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus SYSTEM/*password*@PDB2**

…

SQL>

1. Create a database link in the destination PDB (PDB2) that will connect to the source PDB (PDB1). Refer to *Course Practice Environment: Security Credentials* for the password value.

SQL> **CREATE DATABASE LINK link\_pdb1 CONNECT TO system IDENTIFIED**

**BY *password* USING 'PDB1';**

Database link created. SQL>

1. Drop the target user created in the previous import operation.

SQL> **DROP USER oetest CASCADE;**

User dropped.

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Invoke Data Pump Import and use the NETWORK\_LINK parameter to initiate an import via a database link. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **impdp SYSTEM/*password*@PDB2 SCHEMAS=oe REMAP\_SCHEMA=oe:oetest NETWORK\_LINK=link\_pdb1**

Import: Release 18.0.0.0.0 - Production on Fri Apr 13 12:29:29 2018

Version 18.1.0.0.0

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

Connected to: Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

Starting "SYSTEM"."SYS\_IMPORT\_SCHEMA\_01": SYSTEM/\*\*\*\*\*\*\*\*@PDB2 SCHEMAS=oe REMAP\_SCHEMA=oe:oetest NETWORK\_LINK=link\_pdb1

Estimate in progress using BLOCKS method... Processing object type SCHEMA\_EXPORT/TABLE/TABLE\_DATA Total estimation using BLOCKS method: 128 KB Processing object type SCHEMA\_EXPORT/USER

Processing object type SCHEMA\_EXPORT/SYSTEM\_GRANT Processing object type SCHEMA\_EXPORT/ROLE\_GRANT Processing object type SCHEMA\_EXPORT/DEFAULT\_ROLE

Processing object type SCHEMA\_EXPORT/PRE\_SCHEMA/PROCACT\_SCHEMA Processing object type SCHEMA\_EXPORT/SEQUENCE/SEQUENCE Processing object type SCHEMA\_EXPORT/TABLE/TABLE

. . imported "OETEST"."ORDERS" 105

rows

. . imported "OETEST"."ORDER\_ITEMS" 665

rows

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX Processing object type SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

Processing object type SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

Processing object type SCHEMA\_EXPORT/STATISTICS/MARKER

Job "SYSTEM"."SYS\_IMPORT\_SCHEMA\_01" successfully completed at Fri Apr 13 12:30:26 2018 elapsed 0 00:00:55

[oracle@MYDBCS ~]$

1. Verify that the OE schema was imported as OETEST into PDB2.
   1. Start SQL\*Plus and connect to PDB2 as the OETEST user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus oetest/*password*@PDB2**

… SQL>

* 1. View the list of tables to which the OETEST user has access.

SQL> **SELECT table\_name FROM user\_tables;**

TABLE\_NAME

---------------------------------------------------------------- ORDERS

ORDER\_ITEMS

SQL>

* 1. Query the number of rows in the ORDER\_ITEMS table. The table has 665 rows.

SQL> **SELECT count(\*) FROM order\_items;**

COUNT(\*)

----------

665

SQL>

* 1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Question: What are the advantages and drawbacks of this type of Data Pump import?

Answer: There are no dump files involved. If an import operation is performed over an unencrypted network link, then all data is imported as clear text even if it is encrypted in the database.

**Practice 15-2: Loading Data into a PDB from an External File**

### Overview

In this practice, you use SQL\*Loader to perform the following load operations:

* Load data into the SH.PRODUCTS table in PDB2 by using SQL\*Loader in express mode. Data and control files are provided.
* Load data into the SH.INVENTORIES table in PDB2 by using SQL\*Loader in conventional mode.
* Load data into the SH.INVENTORIES table in PDB2 by using SQL\*Loader in direct mode.

### Assumptions

You are logged in as the oracle user.

### Tasks

##### Load Data by Using SQL\*Loader in Express Mode

As the SH user, use SQL\*Loader in Express Mode to load data from the

$HOME/labs/products.dat data file into the SH.PRODUCTS table in PDB2.

1. If you did not complete Practice 15-1 Moving Data From One PDB to Another, execute the

$HOME/labs/DP\_setup.sh shell script.

$ $HOME/labs/DP\_setup.sh

1. View the products.dat file to learn about its structure.

[oracle@MYDBCS ~]$ **cat /home/oracle/labs/products.dat**

4001,ENG,Door,Outdoor 4002,FRE,Porte,Porte exterieure 4003,SPA,Puerta,Puerta exterior 4004,GER,Tur,Auberliche Tur 5001,ENG,Shutter,Outdoor shutter 5002,FRE,Volet,Volet exterieur 5003,SPA,Obturador,Obturador exterior 5004,GER,Fenster, Fensterladen

[oracle@MYDBCS ~]$

1. Start SQL\*Plus and connect to PDB2 as the SH user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus sh/*password*@PDB2**

…

SQL>

1. Count the number of rows in the SH.PRODUCTS table. The results indicate that there are six rows in the table and, therefore, six products.

SQL> **select count(\*) from sh.products;**

COUNT(\*)

----------

6

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Change to the /home/oracle/labs directory.

[oracle@MYDBCS ~]$ **cd /home/oracle/labs**

1. Start SQL\*Loader, connect to PDB2 as the SH user, and load the records from the products.dat file into the SH.PRODUCTS table in PDB2. The results show that eight rows were successfully loaded. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS labs]$ **sqlldr sh/*password*@PDB2 TABLE=products**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 14:12:23 2018

Version 18.1.0.0.0

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

Express Mode Load, Table: PRODUCTS

Path used: External Table, DEGREE\_OF\_PARALLELISM=AUTO

Table PRODUCTS:

8 Rows successfully loaded.

Check the log files: products.log products\_%p.log\_xt

for more information about the load.

[oracle@MYDBCS labs]$

1. Start SQL\*Plus and connect to PDB2 as the SH user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS labs]$ **sqlplus sh/*password*@PDB2**

…

SQL>

1. Verify that the table is loaded with the eight records from the products.dat file. The results show that the records were loaded.

|  |  |  |  |
| --- | --- | --- | --- |
| PRODUCT\_ID | COU | LABEL | DETAILED\_LABEL |
| ---------- | --- | ---------- | ------------------ |
| 1001 | ENG | Shutter1 | Outdoor shutter1 |
| 1002 | FRE | Porte1 | Porte exterieure1 |
| 1003 | SPA | Puerta1 | Puerta exterior1 |
| 1004 | GER | Tur1 | Auberliche Tur1 |
| 1005 | FRE | Volet1 | Volet exterieur1 |
| 1007 | GER | Fenster1 | Fensterladen1 |
| 4001 | ENG | Door | Outdoor |
| 4002 | FRE | Porte | exterieure |
| 4003 | SPA | Puerta | exterior |
| 4004 | GER | Tur | Auberliche Tur |
| 5001 | ENG | Shutter | Outdoor shutter |
| PRODUCT\_ID | COU | LABEL | DETAILED\_LABEL |
| ---------- | --- | ---------- | ------------------ |
| 5002 | FRE | Volet | exterieur |
| 5003 | SPA | Obturador | exterior |
| 5004 | GER | Fenster | Fensterladen |

1. Exit SQL\*Plus.

SQL> **SELECT \* FROM products;**

14 rows selected.

SQL>

SQL> **EXIT**

…

[oracle@MYDBCS labs]$

1. View the products.log file.

[oracle@MYDBCS labs]$ **cat products.log**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 14:12:23 2018

Version 18.1.0.0.0

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

Express Mode Load, Table: PRODUCTS Data File: products.dat

Bad File: products\_%p.bad Discard File: none specified

(Allow all discards)

Number to load: ALL Number to skip: 0 Errors allowed: 50

Continuation: none specified Path used: External Table

Table PRODUCTS, loaded from every logical record. Insert option in effect for this table: APPEND

Column Name Position Len Term Encl Datatype

------------------------------ ---------- ----- ---- ---- ------ PRODUCT\_ID FIRST \* ,

CHARACTER

COUNTRY NEXT \* ,

CHARACTER

LABEL NEXT \* ,

CHARACTER

DETAILED\_LABEL NEXT \* , CHARACTER

Generated control file for possible reuse: OPTIONS(EXTERNAL\_TABLE=EXECUTE, TRIM=LRTRIM) LOAD DATA

INFILE 'products' APPEND

INTO TABLE PRODUCTS FIELDS TERMINATED BY "," (

PRODUCT\_ID, COUNTRY, LABEL,

DETAILED\_LABEL

)

End of generated control file for possible reuse.

…

Table PRODUCTS:

8 Rows successfully loaded.

Run began on Fri Apr 13 14:12:23 2018

Run ended on Fri Apr 13 14:12:27 2018

Elapsed time was: 00:00:03.51 CPU time was: 00:00:00.33

[oracle@MYDBCS labs]$

1. Question: Which operations did SQL\*Loader execute in express mode?

Answer: SQL\*Loader first created a temporary external table, used the external table to load the content of the external data file into the table, and finally dropped the temporary external table.

1. In the /home/oracle/labs directory where you are working, find the file named products\_nnnn.log\_xt that you just created and display its contents. The date in the file listing will distinguish the right file from the others.

[oracle@MYDBCS labs]$ **ls -l products\_\*.log\_xt**

-rw-r--r-- 1 oracle oinstall 852 Apr 13 14:12 products\_25651.log\_xt

[oracle@MYDBCS labs]$

[oracle@MYDBCS labs]$ cat products\_25651.log\_xt LOG file opened at 04/13/18 14:12:26

Total Number of Files=1 Data File: products.dat

Log File: products\_25651.log\_xt

LOG file opened at 04/13/18 14:12:26 Bad File: products\_25651.bad

Field Definitions for table SYS\_SQLLDR\_X\_EXT\_PRODUCTS

Record format DELIMITED BY NEWLINE

Data in file has same endianness as the platform

Reject rows with all null fields Fields in Data Source:

PRODUCT\_ID CHAR (255)

Terminated by ","

Trim whitespace from left and right COUNTRY CHAR (255)

Terminated by ","

Trim whitespace from left and right LABEL CHAR (255)

Terminated by ","

Trim whitespace from left and right DETAILED\_LABEL CHAR (255)

Terminated by ","

Trim whitespace from left and right

[oracle@MYDBCS labs]$

##### Load Data by Using SQL\*Loader in Conventional Mode

In this section, you will load data into the SH.INVENTORIES table in PDB2 by using SQL\*Loader in conventional mode. Currently, there are 476 rows in the SH.INVENTORIES table.

1. Make sure that your current directory is /home/oracle/labs.

[oracle@MYDBCS ~]$ **cd /home/oracle/labs**

[oracle@MYDBCS labs]$

1. Start SQL\*Plus and connect to PDB2 as the SH user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS labs]$ **sqlplus sh/*password*@PDB2**

…

SQL>

1. Determine the number of rows in the SH.INVENTORIES table. The result shows 476 rows.

SQL> **SELECT count(\*) FROM inventories;**

COUNT(\*)

----------

476

SQL>

1. Exit from SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS labs]$

1. Start SQL\*Loader, connect to PDB2 as the SH user, and load the SH.INVENTORIES table from the $HOME/labs/DP\_inventories.dat data file in conventional mode. Refer to *Course Practice Environment: Security Credentials* for the password value. The result shows that 83 rows were successfully loaded in the SH.INVENTORIES table.

Commit point reached - logical record count 83

Table SH.INVENTORIES:

83 Rows successfully loaded.

Check the log file: inventories.log

for more information about the load.

[oracle@MYDBCS labs]$

Conventional

Path used:

[oracle@MYDBCS labs]$ **sqlldr userid=sh/*password*@PDB2 control=DP\_inventories.ctl log=inventories.log data=DP\_inventories.dat**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 16:27:06 2018

Version 18.1.0.0.0

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

1. Start SQL\*Plus and connect to PDB2 as the SH user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS labs]$ **sqlplus sh/*password*@PDB2**

…

SQL>

1. Determine the number of rows in the SH.INVENTORIES table. The result shows 559 rows.

SQL> **SELECT count(\*) FROM inventories;**

COUNT(\*)

----------

559

SQL>

1. Question: Did SQL\*Loader append new rows or replace rows in the SH.INVENTORIES

table?

Answer: Originally, there were 476 rows in this table. Now there are 559 rows, which means 83 new rows were added, or "appended," by SQL\*Loader.

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS labs]$

1. View the inventories.log file. Notice that the insert option in effect for the

SH.INVENTORIES table is APPEND.

\* ,

FIRST

WAREHOUSE\_ID CHARACTER

------------------------------ ---------- ----- ---- ---- ------

---------------

Position Len Term Encl

Column Name Datatype

$ **cat inventories.log**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 16:52:41 2018

Version 18.1.0.0.0

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

Control File: DP\_inventories.ctl Data File: DP\_inventories.dat Bad File: DP\_inventories.bad

Discard File: none specified (Allow all discards)

Number to load: ALL Number to skip: 0 Errors allowed: 50

Bind array: 250 rows, maximum of 1048576 bytes Continuation: none specified

Path used: Conventional

Table SH.INVENTORIES, loaded from every logical record. Insert option in effect for this table: APPEND

|  |  |  |  |
| --- | --- | --- | --- |
| PRODUCT\_ID CHARACTER |  | NEXT | \* , |
| QUANTITY\_ON\_HAND CHARACTER |  | NEXT | \* , |
| Table SH.INVENTORIES: |  |  |  |
| 83 Rows successfully | loaded. |  |  |
| 0 Rows not loaded due to data errors.  0 Rows not loaded because all WHEN clauses were failed.  0 Rows not loaded because all fields were null.  Space allocated for bind array: 193500 bytes(250 rows)  Read buffer bytes: 1048576  Total logical records skipped: 0  Total logical records read: 83  Total logical records rejected: 0  Total logical records discarded: 0  Run began on Fri Apr 13 16:52:41 2018  Run ended on Fri Apr 13 16:52:43 2018  Elapsed time was: 00:00:01.63 CPU time was: 00:00:00.33  [oracle@MYDBCS labs]$ | | | |

1. View the content of the control file named DP\_inventories.ctl in the vi editor. Notice the APPEND command.

$ vi DP\_inventories.ctl

...

LOAD DATA

infile '/home/oracle/labs/DP\_inventories.dat' INTO TABLE SH.INVENTORIES

**APPEND**

FIELDS TERMINATED BY ','

(warehouse\_id, product\_id, quantity\_on\_hand)

$

1. Change APPEND to TRUNCATE so that the control file truncates the table. Save the file and quit the vi editor (:wq).

...

LOAD DATA

infile '/home/oracle/labs/DP\_inventories.dat' INTO TABLE SH.INVENTORIES

**TRUNCATE**

FIELDS TERMINATED BY ','

(warehouse\_id, product\_id, quantity\_on\_hand)

$

[oracle@MYDBCS labs]$ **sqlldr userid=sh/*password*@PDB2 control=DP\_inventories.ctl log=inventories.log data=DP\_inventories.dat ROWS=10**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 17:00:39 2018

Version 18.1.0.0.0

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

Path used:

Conventional

Table SH.INVENTORIES:

83 Rows successfully loaded.

Check the log file:

1. Start SQL\*Loader, connect to PDB2 as the SH user, and re-execute the load operation with the ROWS parameter set to 10. Refer to *Course Practice Environment: Security Credentials* for the password value.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Commit | point | reached | - logical | record | count | 10 |
| Commit | point | reached | - logical | record | count | 20 |
| Commit | point | reached | - logical | record | count | 30 |
| Commit | point | reached | - logical | record | count | 40 |
| Commit | point | reached | - logical | record | count | 50 |
| Commit | point | reached | - logical | record | count | 60 |
| Commit | point | reached | - logical | record | count | 70 |
| Commit | point | reached | - logical | record | count | 80 |
| Commit | point | reached | - logical | record | count | 83 |

inventories.log

for more information about the load.

[oracle@MYDBCS labs]$

1. Start SQL\*Plus and connect to PDB2 as the SH user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus sh/*password*@PDB2**

…

SQL>

1. Verify the number of rows in the INVENTORIES table. The table now has 83 rows. The

TRUNCATE option cleared out the original rows in the table and inserted 83 new rows.

SQL> **SELECT count(\*) FROM inventories;**

COUNT(\*)

----------

83

SQL>

##### Re-enable the Check Constraint

Suppose a DBA discovers that the CHECK constraint was disabled on the WAREHOUSE\_ID column in the SH.INVENTORIES table at the time of the load. This disabled constraint allowed only values within a certain range. Use the $HOME/labs/DP\_check.sql SQL script to empty the table and re-enable the check constraint. Then reload the table.

1. Execute the $HOME/labs/DP\_check.sql script.

SQL> **@$HOME/labs/DP\_check.sql**

Connected.

Table truncated.

Table altered.

Disconnected from Oracle Database 18c EE High Perf Release

18.0.0.0.0 - Production

Version 18.1.0.0.0 [oracle@MYDBCS labs]$

1. Start SQL\*Loader, connect to PDB2 as the SH user, and reload the table. Refer to *Course Practice Environment: Security Credentials* for the password value. The results indicate that 20 rows were successfully loaded into the SH.INVENTORIES table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Path used: | Conventional |  | | |
| Commit point | reached - logical | record | count | 10 |
| Commit point | reached - logical | record | count | 20 |
| Commit point | reached - logical | record | count | 30 |
| Commit point | reached - logical | record | count | 40 |
| Commit point | reached - logical | record | count | 50 |
| Commit point | reached - logical | record | count | 60 |
| Commit point | reached - logical | record | count | 70 |
| Commit point | reached - logical | record | count | 80 |

1. View the inventories.log file. The log file says that 20 rows were successfully loaded into the SH.INVENTORIES table; however, 51 rows were not loaded due to errors.

$ **cat inventories.log**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 17:06:03 2018

Version 18.1.0.0.0

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

Control File: DP\_inventories.ctl Data File: DP\_inventories.dat Bad File: DP\_inventories.bad

Discard File: none specified

[oracle@MYDBCS labs]$ **sqlldr userid=sh/*password*@PDB2 control=DP\_inventories.ctl log=inventories.log data=DP\_inventories.dat ROWS=10**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 17:06:03 2018

Version 18.1.0.0.0

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

Table SH.INVENTORIES:

20 Rows successfully loaded.

Check the log file: inventories.log

for more information about the load.

[oracle@MYDBCS labs]$

(Allow all discards)

Number to load: ALL Number to skip: 0 Errors allowed: 50

Bind array: 10 rows, maximum of 1048576 bytes Continuation: none specified

Path used: Conventional

Table SH.INVENTORIES, loaded from every logical record. Insert option in effect for this table: APPEND

Column Name Position Len Term Encl Datatype

------------------------------ ---------- ----- ---- ---- ------

---------------

WAREHOUSE\_ID FIRST \* , CHARACTER

PRODUCT\_ID NEXT \* ,

CHARACTER

QUANTITY\_ON\_HAND NEXT \* , CHARACTER

Record 21: Rejected - Error on table SH.INVENTORIES.

ORA-02290: check constraint (SH.CK\_WAREHOUSE\_ID) violated

Record 22: Rejected - Error on table SH.INVENTORIES.

ORA-02290: check constraint (SH.CK\_WAREHOUSE\_ID) violated

…

Record 70: Rejected - Error on table SH.INVENTORIES.

ORA-02290: check constraint (SH.CK\_WAREHOUSE\_ID) violated

Record 71: Rejected - Error on table SH.INVENTORIES.

ORA-02290: check constraint (SH.CK\_WAREHOUSE\_ID) violated

MAXIMUM ERROR COUNT EXCEEDED - Above statistics reflect partial run.

Table SH.INVENTORIES:

20 Rows successfully loaded.

51 Rows not loaded due to data errors.

0 Rows not loaded because all WHEN clauses were failed.

Read buffer bytes: 1048576

Total logical records skipped: 0

Total logical records read: 80

Total logical records rejected: 51

Total logical records discarded: 0

Run began on Fri Apr 13 17:06:03 2018

Run ended on Fri Apr 13 17:06:04 2018

Elapsed time was: 00:00:01.00 CPU time was: 00:00:00.33

[oracle@MYDBCS labs]$

7740 bytes(10

Space allocated for bind array: rows)

0 Rows not loaded because all fields were null.

1. Question: Did SQL\*Loader try to load all rows?

Answer: No. After 20 rows successfully loaded, 51 rows did not load due to a constraint violation error. The load stopped at this point. The default number of errors tolerated is 50. When the number was exceeded, SQL\*Loader stopped.

##### Load Data by using SQL\*Loader in Direct Mode

Observe how SQL\*Loader behaves when loading the SH.INVENTORIES table in direct mode.

1. Start SQL\*Loader, connect to PDB2 as the SH user, and load the SH.INVENTORIES table in direct mode. Refer to *Course Practice Environment: Security Credentials* for the password value. The results indicate that the load completed and the record count is 83.

Save data point reached - logical record count 10. Save data point reached - logical record count 20.

Save data point reached - logical record count 30.

Direct

Path used:

[oracle@MYDBCS labs]$ **sqlldr userid=sh/*password*@PDB2 control=DP\_inventories.ctl log=inventories.log data=DP\_inventories.dat ROWS=10 DIRECT=TRUE**

SQL\*Loader: Release 18.0.0.0.0 - Production on Fri Apr 13 17:09:24 2018

Version 18.1.0.0.0

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

Save data point reached - logical record count 40. Save data point reached - logical record count 50. Save data point reached - logical record count 60. Save data point reached - logical record count 70. Save data point reached - logical record count 80.

Load completed - logical record count 83. Table SH.INVENTORIES:

83 Rows successfully loaded.

Check the log file: inventories.log

for more information about the load.

[oracle@MYDBCS labs]$

1. Question: Does the direct load use the SQL INSERT statement? How does the direct path commit the rows inserted?

Answer: The direct load loads records into the blocks, writing the data blocks directly to the database files. You can observe that there is no COMMIT instruction, but SAVE instead.

During a data save, only full database blocks are written to the database.

1. Question: Did it enforce the CHECK constraint?

Answer: No, it did not. This is the reason all rows were loaded, regardless of the

WAREHOUSE\_ID value to be inserted.

1. Question: Does SQL\*Loader in direct mode ignore all constraints?

Answer: No, it does not. It enforces PRIMARY KEY, UNIQUE, and NOT NULL constraints.

1. Disconnect from the compute node and close the terminal window.

**Practices for Lesson 16: Backup and Recovery**

**Concepts**

**Practices for Lesson 16**

There are no practices for this lesson.

**Practices for Lesson 17: Backup and Recovery**

**Configuration**

**Practices for Lesson 17: Overview**

### Overview

In these practices, you learn how to configure your database to enable recovery from various losses. You verify the control file configuration, the Fast Recovery Area (FRA), redo log groups, ARCHIVELOG mode, and redundant archive log destinations.

How to configure your database for recovery:

* + Ensure redundancy of control files. If a control file is damaged or lost, recovery is easier if you have another copy.
  + Review the fast recovery area configuration.
  + Ensure that there are at least two redo log members in each group. If a redo log member is damaged or lost, recovery is easier when you have an additional member in the group.
  + Place your database in ARCHIVELOG mode. In all cases, you will be able to recover the database either completely or incompletely depending on which database files have been damaged or lost.
  + Configure redundant archive log destinations. In cases where you lost archive log files and you need them to recover the database, you will be able to perform an incomplete recovery, unless you have a duplicate version of the archive log in another destination.

**Practice 17-1: Verifying that the Control File is Multiplexed**

### Overview

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

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

### Assumptions

You are logged in as the oracle user.

### Tasks

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

oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

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

$ **sqlplus / AS SYSDBA**

…

SQL>

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

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

NAME

----------------------------------------------------------------

/u02/app/oracle/oradata/ORCL/control01.ctl

/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl

SQL>

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

1. View the CONTROL\_FILES parameter. Notice that the paths to the control files are stored in this parameter. The results below are formatted for easier viewing.

string

TYPE

-------

NAME

---------------------------------------------------- VALUE

--------------------------------------------------------

control\_files

/u02/app/oracle/oradata/ORCL/control01.ctl,

/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl

SQL>

SQL> **SHOW PARAMETER control\_files**

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

SQL> **CREATE PFILE FROM SPFILE;**

File created. SQL>

1. Shut down the database instance in IMMEDIATE mode.

SQL> **SHUTDOWN IMMEDIATE**

Database closed. Database dismounted.

ORACLE instance shut down.

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

1. Create a directory for the new control file.

[oracle@MYDBCS ~]$ **mkdir -p**

**/u01/app/oracle/controlfiles\_dir/ORCL**

[oracle@MYDBCS ~]$

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

[oracle@MYDBCS ~]$ **cp $ORACLE\_HOME/dbs/initORCL.ora**

**$ORACLE\_HOME/dbs/backup\_initORCL.ora**

[oracle@MYDBCS ~]$

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

[oracle@MYDBCS ~]$ **cp /u02/app/oracle/oradata/ORCL/control01.ctl**

**/u01/app/oracle/controlfiles\_dir/ORCL/control03.ctl**

[oracle@MYDBCS ~]$

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

$ **vi $ORACLE\_HOME/dbs/initORCL.ora**

...

\*.control\_files='/u02/app/oracle/oradata/ORCL/control01.ctl', '/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl',

**'/u01/app/oracle/controlfiles\_dir/ORCL/control03.ctl'**

...

$

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

privilege. You are connected to an idle instance.

$ **sqlplus / AS SYSDBA**

…

SQL>

1. Start the database instance.

SQL> **STARTUP**

ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes Variable Size 704643072 bytes

Database Buffers 1979711488 bytes Redo Buffers 74985472 bytes Database mounted.

Database opened.

SQL>

1. View the CONTROL\_FILES parameter again.

------------------------------------ ----------- VALUE

------------------------------

control\_files string

/u02/app/oracle/oradata/ORCL/control01.ctl,

/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl

TYPE

SQL> **SHOW PARAMETER control\_files**

NAME

SQL>

1. Question: Why does the CONTROL\_FILES parameter still show only two control files? Answer: By default, the database instance starts up with the SPFILE. If an SPFILE does not exist, then the instance starts up with a PFILE. In this case, both an SPFILE and PFILE are

present, so the SPFILE takes precedence. You configured the PFILE, not the SPFILE. The

SPFILE still contains only two references.

1. Re-create the third control file because the current version is no longer an exact copy of the others.
   1. Shut down the database instance with the IMMEDIATE option.

SQL> **SHUTDOWN IMMEDIATE**

Database closed. Database dismounted.

ORACLE instance shut down.

SQL>

* 1. Exit SQL\*Plus.

SQL> **EXIT**

* 1. Use the cp command to re-create control03.ctl.

[oracle@MYDBCS ~]$ **cp /u02/app/oracle/oradata/ORCL/control01.ctl**

**/u01/app/oracle/controlfiles\_dir/ORCL/control03.ctl**

[oracle@MYDBCS ~]$

1. Re-create the SPFILE from the updated PFILE.
   1. Start SQL\*Plus and connect to the CDB root as the SYS user with the SYSDBA

privilege. You are connected to an idle instance.

[oracle@MYDBCS ~]$ **sqlplus / AS SYSDBA**

… SQL>

* 1. Create the SPFILE.

SQL> **CREATE SPFILE FROM PFILE;**

File created. SQL>

1. Start the database instance.

SQL> **STARTUP**

ORACLE instance started.

Total System Global Area 2768239832 bytes Fixed Size 8899800 bytes Variable Size 704643072 bytes

Database Buffers 1979711488 bytes

74985472 bytes

Redo Buffers Database mounted. Database opened.

SQL>

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

------------------------------------ ----------- VALUE

------------------------------

control\_files string

/u02/app/oracle/oradata/ORCL/control01.ctl,

/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl,

/u01/app/oracle/controlfiles\_dir/ORCL/control03.ctl

SQL>

TYPE

SQL> **SHOW PARAMETER control\_files**

NAME

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

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

NAME

----------------------------------------------------------------

/u02/app/oracle/oradata/ORCL/control01.ctl

/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl

/u01/app/oracle/controlfiles\_dir/ORCL/control03.ctl

SQL>

**Practice 17-2: Checking Storage Availability**

### Overview

In this practice, you verify that there is enough room on /u03 for files that you create in this lesson and the next.

### Assumptions

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

### Tasks

1. In your terminal window, check the use of disk storage.

[oracle@MYDBCS ~]$ **df –h**

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/redoVolGroup-lvol0

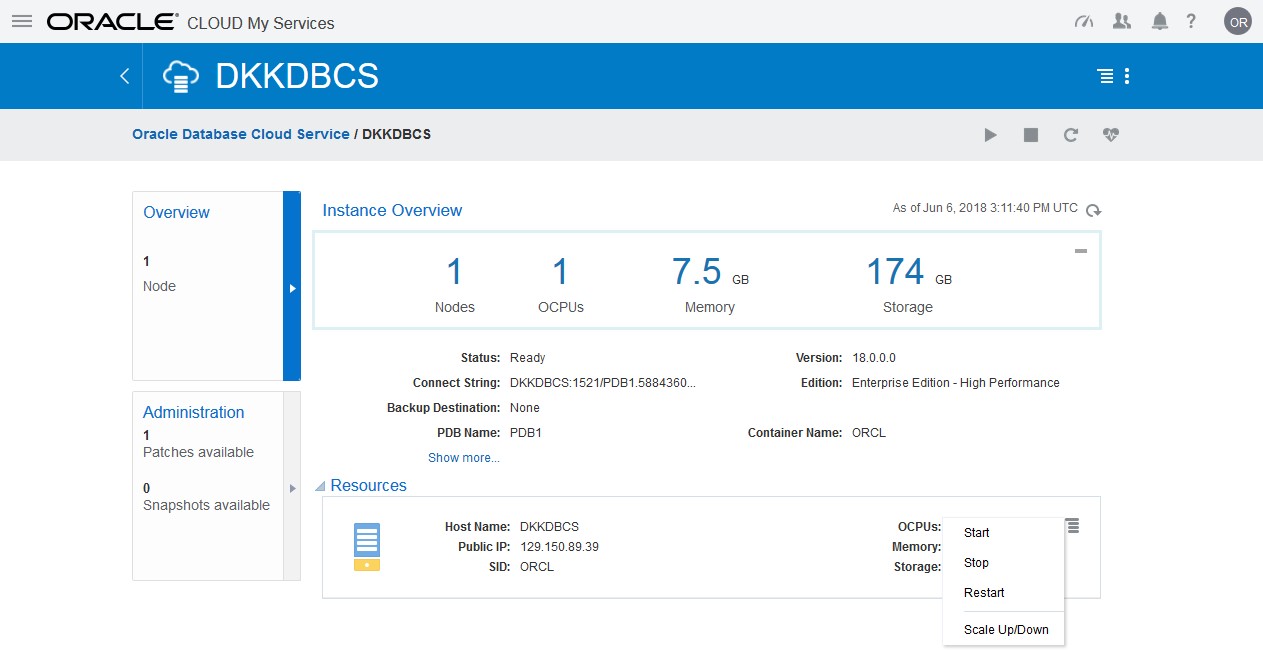
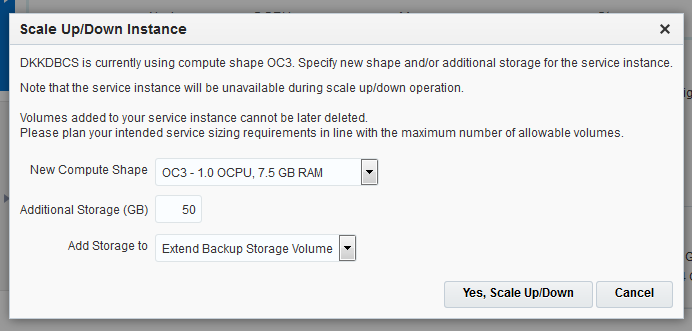
25G 3.3G 20G 14% /u04

[oracle@MYDBCS ~]$

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| /dev/mapper/vg\_main-lv\_root | | | | | |
|  | 25G | 3.7G | 21G | 16% | / |
| tmpfs | 3.7G | 0 | 3.7G | 0% | /dev/shm |
| /dev/xvdb1 | 477M | 70M | 379M | 16% | /boot |
| /dev/xvde1 | 59G | 9.7G | 47G | 18% | /u01 |
| /dev/mapper/dataVolGroup-lvol0 | | | | | |
| 50G | | 7.8G | 39G | 17% | /u02 |
| /dev/mapper/fraVolGroup-lvol0 | | | | | |
| 6.8G | | 5.6G | 854M | 87% | /u03 |

1. If the used storage (Use%) on /u03 is more than 50%, add storage to the database deployment.
   1. Close the connection to the compute node.
   2. Open the browser and access the Oracle Cloud web site by entering

https://cloud.oracle.com/home.

* 1. Click **Sign In**.
  2. On the Cloud Account page, select Cloud Account with Identity Cloud Service in the menu and enter the account name as provided. Click **My Services**.
  3. On the Oracle Cloud Account Sign In page, enter the user name and password as provided. Click **Sign In**.
  4. Expand the menu next to Oracle Cloud My Services.
  5. Expand the **Services** menu and click **Database**.
  6. Search for your database deployment (instance) by entering its name in the search field.
  7. Click the link on the database deployment (instance) name.
  8. Expand the menu in the Resources area and select **Scale Up/Down**.
  9. In the Scale Up/Down Instance window, enter **50** in the “Additional Storage (GB)” field and select **Extend Backup Storage Volume** in the “Add Storage to” menu.
  10. Click **Yes, Scale Up/Down**.
  11. A confirmation message is displayed. Periodically refresh the page until you see that the status is once again Ready.
  12. Sign out of your Cloud account.

1. Return to your terminal window and connect to the compute node.

oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. In your terminal window, check the use of disk storage again to ensure you now have room for backups.

[oracle@MYDBCS ~]$ **df –h**

Filesystem Size Used Avail Use% Mounted on

56G 6.3G 47G 12% /u03

[oracle@MYDBCS ~]$

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| /dev/mapper/vg\_main-lv\_root | | | | | |
|  | 25G | 3.7G | 21G | 16% | / |
| tmpfs | 3.7G | 0 | 3.7G | 0% | /dev/shm |
| /dev/xvdb1 | 477M | 70M | 379M | 16% | /boot |
| /dev/xvde1 | 59G | 9.7G | 47G | 18% | /u01 |
| /dev/mapper/dataVolGroup-lvol0 | | | | | |
| 50G | | 7.8G | 39G | 17% | /u02 |
| /dev/mapper/redoVolGroup-lvol0 | | | | | |
| 25G | | 3.3G | 20G | 14% | /u04 |
| /dev/mapper/fraVolGroup-lvol0 | | | | | |

**Practice 17-3: Configuring the Size of the Fast Recovery Area**

### Overview

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

### Assumptions

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

### Tasks

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

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

1. View the values of the DB\_RECOVERY\_FILE\_DEST and DB\_RECOVERY\_FILE \_ DEST\_SIZE initialization parameters.

big integer

string

db\_recovery\_file\_dest

/u03/app/oracle/fast\_recovery\_area db\_recovery\_file\_dest\_size

4G

SQL>

------------------------------------ ----------- VALUE

------------------------------

TYPE

NAME

SQL> **SHOW PARAMETER db\_recovery\_file\_dest**

1. Question: Is the fast recovery area enabled?

Answer: Yes. The DB\_RECOVERY\_FILE\_DEST and DB\_RECOVERY\_FILE\_DEST\_SIZE

parameters values are not null, indicating that the fast recovery area is enabled.

1. Question: Which changes can you make to the fast recovery area? Answer: You can change the location and size for the fast recovery area.
2. Question: Does changing the size of the fast recovery area require the database to be restarted?

Answer: No, a restart is not required for this change because the

DB\_RECOVERY\_FILE\_DEST\_SIZE parameter is dynamic.

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

SQL> **ALTER SYSTEM SET db\_recovery\_file\_dest\_size = 12G SCOPE=both;**

System altered.

SQL>

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

1. View the DB\_RECOVERY\_FILE\_DEST\_SIZE initialization parameter again. The result verifies that the size has been set to 12GB.

VALUE

------

12G

big integer

db\_recovery\_file\_dest\_size

SQL>

-------------------------------- -----------

TYPE

NAME

SQL> **SHOW PARAMETER db\_recovery\_file\_dest\_size**

**Practice 17-4: Verifying that the Redo Log File is Multiplexed**

### Overview

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

### Assumptions

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

### Tasks

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

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

GROUP# STATUS MEMBER

------- ------- ----------------------------------------

3 /u04/app/oracle/redo/redo03.log

2 /u04/app/oracle/redo/redo02.log

1 /u04/app/oracle/redo/redo01.log

SQL>

1. Question: Why is it recommended to have three groups when two would be sufficient? Answer: The Oracle Database server treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next.

After all groups have been written to, the Oracle Database server begins overwriting

information in the first log group. If the database is configured in ARCHIVELOG mode, the LGWR cannot overwrite data in the first log group if it has not been archived.

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

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

1. Add another member to each redo log group. Name each member as redo*nn*b.log where

*nn* represents the group number.

SQL> **ALTER DATABASE ADD LOGFILE MEMBER**

**'/u03/app/oracle/fast\_recovery\_area/ORCL/redo01b.log' TO GROUP 1;**

Database altered.

SQL> **ALTER DATABASE ADD LOGFILE MEMBER**

**'/u03/app/oracle/fast\_recovery\_area/ORCL/redo02b.log' TO GROUP 2;**

Database altered.

SQL> **ALTER DATABASE ADD LOGFILE MEMBER**

**'/u03/app/oracle/fast\_recovery\_area/ORCL/redo03b.log' TO GROUP 3;**

Database altered. SQL>

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

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

GROUP# STATUS MEMBER

------- ------- ------------------------------------------------

1 INVALID

/u03/app/oracle/fast\_recovery\_area/ORCL/redo01b.log

1. /u04/app/oracle/redo/redo01.log
2. INVALID

/u03/app/oracle/fast\_recovery\_area/ORCL/redo02b.log

1. /u04/app/oracle/redo/redo02.log
2. INVALID

/u03/app/oracle/fast\_recovery\_area/ORCL/redo03b.log

3 /u04/app/oracle/redo/redo03.log

6 rows selected.

SQL>

1. Switch the log files and observe the changes.
   1. Find out which log group is the current log group. In this example, the query result shows that group 3 is the current group. Your current group may be different.

SQL>

CURRENT

2 NO

2 YES INACTIVE

2 YES INACTIVE

1

2

3

------- ---------- --- ----------------

MEMBERS ARC STATUS

GROUP#

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

* 1. Switch the log files three times.

SQL> **ALTER SYSTEM SWITCH LOGFILE;**

System altered.

SQL> **ALTER SYSTEM SWITCH LOGFILE;**

System altered.

SQL> **ALTER SYSTEM SWITCH LOGFILE;**

System altered.

SQL>

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

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

6 rows selected.

SQL>

|  |  |  |
| --- | --- | --- |
| GROUP# | STATUS | MEMBER |
| ------- | ------- | ---------------------------------------- |
| 1 |  | /u03/app/oracle/fast\_recovery\_area/ORCL/redo01b.log |
| 1 |  | /u04/app/oracle/redo/redo01.log |
| 2 |  | /u03/app/oracle/fast\_recovery\_area/ORCL/redo02b.log |
| 2 |  | /u04/app/oracle/redo/redo02.log |
| 3 |  | /u03/app/oracle/fast\_recovery\_area/ORCL/redo03b.log |
| 3 |  | /u04/app/oracle/redo/redo03.log |

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

SQL>

CURRENT

2 NO

2 YES INACTIVE

2 YES INACTIVE

1

2

3

------- ---------- --- ----------------

MEMBERS ARC STATUS

GROUP#

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

* 1. Switch the log file.

SQL> **ALTER SYSTEM SWITCH LOGFILE;**

System altered.

SQL>

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

SQL>

2 YES INACTIVE

2 YES ACTIVE

CURRENT

2 NO

1

2

3

------- ---------- --- ----------------

MEMBERS ARC STATUS

GROUP#

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

* 1. Switch the log file again.

SQL> **ALTER SYSTEM SWITCH LOGFILE;**

System altered.

SQL>

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

SQL>

2 YES ACTIVE

CURRENT

2 NO

2 YES ACTIVE

1

2

3

------- ---------- --- ----------------

MEMBERS ARC STATUS

GROUP#

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

* 1. Question: Can the LGWR background process write to only one member of the

CURRENT group in case the other members are missing or damaged?

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

1. To save space in your course practice environment, drop the redo log file members you created in step 4.
   1. Determine which redo log group is current. You cannot drop a member of the current group.

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

GROUP# STATUS

---------- ----------------

1. INACTIVE
2. CURRENT
3. INACTIVE

SQL>

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

SQL> **ALTER DATABASE DROP LOGFILE MEMBER**

**'/u03/app/oracle/fast\_recovery\_area/ORCL/redo01b.log';**

Database altered.

SQL> **alter system switch logfile;**

System altered.

SQL>

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

SQL> **ALTER DATABASE DROP LOGFILE MEMBER**

**'/u03/app/oracle/fast\_recovery\_area/ORCL/redo02b.log';**

Database altered.

SQL> **alter system switch logfile;**

System altered.

SQL>

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

SQL> **ALTER DATABASE DROP LOGFILE MEMBER**

**'/u03/app/oracle/fast\_recovery\_area/ORCL/redo03b.log';**

Database altered.

SQL> **alter system switch logfile;**

System altered.

SQL>

* 1. Verify that each group now has only one member.

SQL>

1 YES ACTIVE

CURRENT

1 NO

1 YES ACTIVE

1

2

3

---------- ---------- --- ----------------

MEMBERS ARC STATUS

GROUP#

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

* 1. Exit from SQL\*Plus.

SQL> **exit**

…

[oracle@MYDBCS ~]$

* 1. Remove the physical files from the operating system.

[oracle@MYDBCS ~]$ **rm**

**/u03/app/oracle/fast\_recovery\_area/ORCL/redo\*.log**

* 1. Verify that the redo log files have been removed.

[oracle@MYDBCS ~]$ **ls /u03/app/oracle/fast\_recovery\_area/ORCL**

archivelog autobackup control02.ctl flashback onlinelog

**Practice 17-5: Verifying that ARCHIVELOG Mode is Configured**

### Overview

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

### Assumptions Tasks

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

[oracle@MYDBCS ~]$ **sqlplus / AS SYSDBA**

…

SQL

1. Issue the ARCHIVE LOG LIST command to verify that the database is in ARCHIVELOG

mode. The result confirms that the database log mode is set to ARCHIVELOG mode.

SQL> **archive log list**

Database log mode Archive Mode

Automatic archival Enabled

Archive destination USE\_DB\_RECOVERY\_FILE\_DEST Oldest online log sequence 27

Next log sequence to archive 29

Current log sequence 29

SQL>

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

1. Exit from SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

**Practices for Lesson 18:**

**Creating Database Backups**

**Practices for Lesson 18: Overview**

### Overview

In these practices, you will create a script file that can be used to re-create the control file. You will also create a whole database backup and a partial database backup.

**Practice 18-1: Backing up the Control File**

### Overview

In this practice, you back up your control file to a trace file and then create a file of SQL commands that can be used to re-create the control file.

##### Tip

The loss of a single control file causes the database instance to fail because all control files must be available at all times. However, recovery can be a simple matter of copying one of the other control files. The loss of all control files is slightly more difficult to recover from, but is not usually catastrophic as long as you created a copy of the control file by backing it up to a trace file.

### Assumptions

You are logged in as the oracle user.

You completed all of the practices in Lesson 17.

### Tasks

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Start SQL\*Plus and connect to the CDB root as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus system/*password***

… SQL>

1. Verify that the control files are multiplexed.

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

NAME

----------------------------------------------------------------

/u02/app/oracle/oradata/ORCL/control01.ctl

/u03/app/oracle/fast\_recovery\_area/ORCL/control02.ctl

/u01/app/oracle/controlfiles\_dir/ORCL/control03.ctl

SQL>

1. Back up the control file to a trace file.

SQL> **ALTER DATABASE BACKUP controlfile TO trace;**

Database altered. SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Navigate to the directory that contains the alert log file and trace files.

[oracle@MYDBCS ~]$ **cd /u01/app/oracle/diag/rdbms/orcl/ORCL/trace**

[oracle@MYDBCS trace]$

1. List the files in this directory. Notice that the directory contains the alert log (alert\_ORCL.log) and many trace files (.trc).

[oracle@MYDBCS trace]$ **ls**

alert\_ORCL.log ORCL\_dbrm\_9261.trc ORCL\_ora\_27453.trc drcORCL.log ORCL\_dbrm\_9261.trm ORCL\_ora\_27453.trm [oracle@MYDBCS trace]$

1. View the end of the alert log and make note of the last trace file created as a backup for the control file. In this example, it is ORCL\_ora\_27453.trc. Your file name will be different.

$ **tail alert\_ORCL.log**

2018-04-09T15:54:42.466903-04:00

alter database backup controlfile to trace 2018-04-09T15:54:42.476498-04:00

Backup controlfile written to trace file

/u01/app/oracle/diag/rdbms/orcl/ORCL/trace/ORCL\_ora\_27453.trc

Completed: alter database backup controlfile to trace [oracle@MYDBCS trace]$

1. View the content of the last generated trace file by using the cat command. Make sure to substitute the name of the trace file with your trace file name.

Between the lines " -- Set #1. NORESETLOGS case" and " -- Set #2. RESETLOGS case", select the code from STARTUP NOMOUNT to ALTER SESSION SET CONTAINER = CDB$ROOT; and copy it to the clipboard.

**Note:** The file names in your database will likely differ from the file names shown in this example.

$ **cat**

##### /u01/app/oracle/diag/rdbms/orcl/ORCL/trace/ORCL\_ora\_27453.trc

…

STARTUP NOMOUNT

CREATE CONTROLFILE REUSE DATABASE "ORCL" NORESETLOGS FORCE LOGGING ARCHIVELOG

MAXLOGFILES 16

MAXLOGMEMBERS 3

MAXDATAFILES 1024

MAXINSTANCES 8

MAXLOGHISTORY 292 LOGFILE

GROUP 1 '/u04/app/oracle/redo/redo01.log' SIZE 1024M BLOCKSIZE 512,

GROUP 2 '/u04/app/oracle/redo/redo02.log' SIZE 1024M BLOCKSIZE 512,

GROUP 3 '/u04/app/oracle/redo/redo03.log' SIZE 1024M BLOCKSIZE 512

-- STANDBY LOGFILE DATAFILE

'/u02/app/oracle/oradata/ORCL/system01.dbf', '/u02/app/oracle/oradata/ORCL/sysaux01.dbf', '/u02/app/oracle/oradata/ORCL/undotbs01.dbf', '/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf', '/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf', '/u02/app/oracle/oradata/ORCL/users01.dbf', '/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf', '/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf', '/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf', '/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf', '/u02/app/oracle/oradata/ORCL/PDB1/PDB1\_users01.dbf', '/u02/app/oracle/oradata/ORCL/PDB2/system01.dbf', '/u02/app/oracle/oradata/ORCL/PDB2/sysaux01.dbf', '/u02/app/oracle/oradata/ORCL/PDB2/undotbs01.dbf'

CHARACTER SET AL32UTF8

;

-- Configure RMAN configuration record 1 VARIABLE RECNO NUMBER;

EXECUTE :RECNO := SYS.DBMS\_BACKUP\_RESTORE.SETCONFIG('BACKUP OPTIMIZATION','ON');

-- Configure RMAN configuration record 2 VARIABLE RECNO NUMBER;

EXECUTE :RECNO := SYS.DBMS\_BACKUP\_RESTORE.SETCONFIG('CONTROLFILE

AUTOBACKUP','ON');

-- Commands to re-create incarnation table

-- Below log names MUST be changed to existing filenames on

-- disk. Any one log file from each branch can be used to

-- re-create incarnation records.

-- ALTER DATABASE REGISTER LOGFILE

'/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_09/o 1\_mf\_1\_1\_%u\_.arc';

-- ALTER DATABASE REGISTER LOGFILE

'/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_09/o 1\_mf\_1\_1\_%u\_.arc';

-- ALTER DATABASE REGISTER LOGFILE

'/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_09/o 1\_mf\_1\_1\_%u\_.arc';

-- Recovery is required if any of the datafiles are restored backups,

-- or if the last shutdown was not normal or immediate. RECOVER DATABASE

-- Block change tracking was enabled, so re-enable it now. ALTER DATABASE ENABLE BLOCK CHANGE TRACKING

USING FILE

'/u02/app/oracle/oradata/ORCL/changetracking/o1\_mf\_fdf3bxbb\_.chg ' REUSE;

-- All logs need archiving and a log switch is needed. ALTER SYSTEM ARCHIVE LOG ALL;

-- Database can now be opened normally.

ALTER DATABASE OPEN;

-- Open all the PDBs.

ALTER PLUGGABLE DATABASE ALL OPEN;

-- Commands to add tempfiles to temporary tablespaces.

-- Online tempfiles have complete space information.

-- Other tempfiles may require adjustment. ALTER TABLESPACE TEMP ADD TEMPFILE

'/u04/app/oracle/oradata/temp/temp01.dbf'

SIZE 136314880 REUSE AUTOEXTEND ON NEXT 655360 MAXSIZE 32767M;

ALTER SESSION SET CONTAINER = PDB$SEED; ALTER TABLESPACE TEMP ADD TEMPFILE

'/u04/app/oracle/oradata/temp/pdbseed\_temp012018-02-08\_13-49-27- 256-PM.dbf'

SIZE 65011712 REUSE AUTOEXTEND ON NEXT 655360 MAXSIZE 32767M;

ALTER SESSION SET CONTAINER = PDB1; ALTER TABLESPACE TEMP ADD TEMPFILE

'/u04/app/oracle/oradata/temp/temp012018-02-08\_13-49-27-256-

PM.dbf'

ALTER SESSION SET CONTAINER = CDB$ROOT;

…

[oracle@MYDBCS trace]$

MAXSIZE

REUSE AUTOEXTEND ON NEXT 655360

SIZE 65011712 32767M;

ALTER SESSION SET CONTAINER = PDB2; ALTER TABLESPACE TEMP ADD TEMPFILE

'/u04/app/oracle/oradata/temp/PDB2/pdbseed\_temp012018-02-08\_13-

49-27-256-PM.dbf'

MAXSIZE

REUSE AUTOEXTEND ON NEXT 655360

SIZE 65011712 32767M;

1. Open an editor and paste the code into a new file named ControlFileBackup.sql in the /home/oracle directory.
2. Question: Which command would allow the recreation of the control files in case of a complete loss of the control files?

Answer: In the case where all control files are lost, the CREATE CONTROLFILE command in the trace file would recreate the missing control files with the right information, keeping the database file structure in terms of data files, redo log files, and other database attributes (ARCHIVELOG, maximum settings).

1. Question: How would you execute the command?

Answer: After trimming the trace file by keeping all commands from the STARTUP NOMOUNT up to ALTER SESSION SET CONTAINER = CDB$ROOT;, you would execute the file as an SQL script.

1. Question: Are the data files, temp files, and control files that structure the ORCL database included in the SQL script?

Answer: Yes, they are included. All data and temp files of the different containers (the CDB root, CDB seed, PDB1, and so on) and the multiplexed redo log files are present.

1. Question: Which other attributes structure the ORCL database?

Answer: The ARCHIVELOG mode, the character set, and the name of the CDB.

1. Question: Why are there two cases—Set #1. NORESETLOGS and Set #2. RESETLOGS?

Answer: The first case from the STARTUP NOMOUNT to the ALTER SESSION SET CONTAINER = CDB$ROOT provides a script to execute a complete database recovery. Use this only if the current versions of all online logs are available. The second case provides a script to execute an incomplete database recovery. Use this only if online logs are damaged. The contents of online logs will be lost, and all backups will be invalidated.

1. Question: When would you have to regenerate the trace file from the current control files?

Answer: Because the control file changes after each data file or redo log file change (adding, removing, resizing) or database attribute change (ARCHIVELOG), you would have to redo the backup of your control file to a trace file.

**Practice 18-2: Verifying Automatic Backups of the Control File and SPFILE**

### Overview

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

### Assumptions

You are logged in to the compute node as the oracle user. You completed the practices in Lesson 17.

### Tasks

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

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

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

Version 18.1.0.0.0

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

connected to target database: ORCL (DBID=2299035716)

RMAN>

1. Show all RMAN settings. Notice the CONFIGURE CONTROLFILE AUTOBACKUP ON;

setting.

RMAN> **SHOW ALL;**

using target database control file instead of recovery catalog RMAN configuration parameters for database with db\_unique\_name ORCL are:

CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default CONFIGURE BACKUP OPTIMIZATION ON;

CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default CONFIGURE CONTROLFILE AUTOBACKUP ON;

CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO

'%F'; # default

CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO

BACKUPSET; # default

CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #

default

CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; #

default

CONFIGURE MAXSETSIZE TO UNLIMITED; # default CONFIGURE ENCRYPTION FOR DATABASE OFF; # default CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default

CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; # default

CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default

CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default CONFIGURE SNAPSHOT CONTROLFILE NAME TO

'/u01/app/oracle/product/18.0.0/dbhome\_1/dbs/snapcf\_ORCL.f'; #

default

RMAN>

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

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

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

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

1. Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete. [oracle@MYDBCS ~]$

**Practice 18-3: Checking Storage Availability**

### Overview

In this practice, you verify that there is enough storage on /u03 for backups you will take in the practices which follow.

### Assumptions

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

### Tasks

1. In your terminal window, check the use of disk storage.

[oracle@MYDBCS ~]$ **df -h**

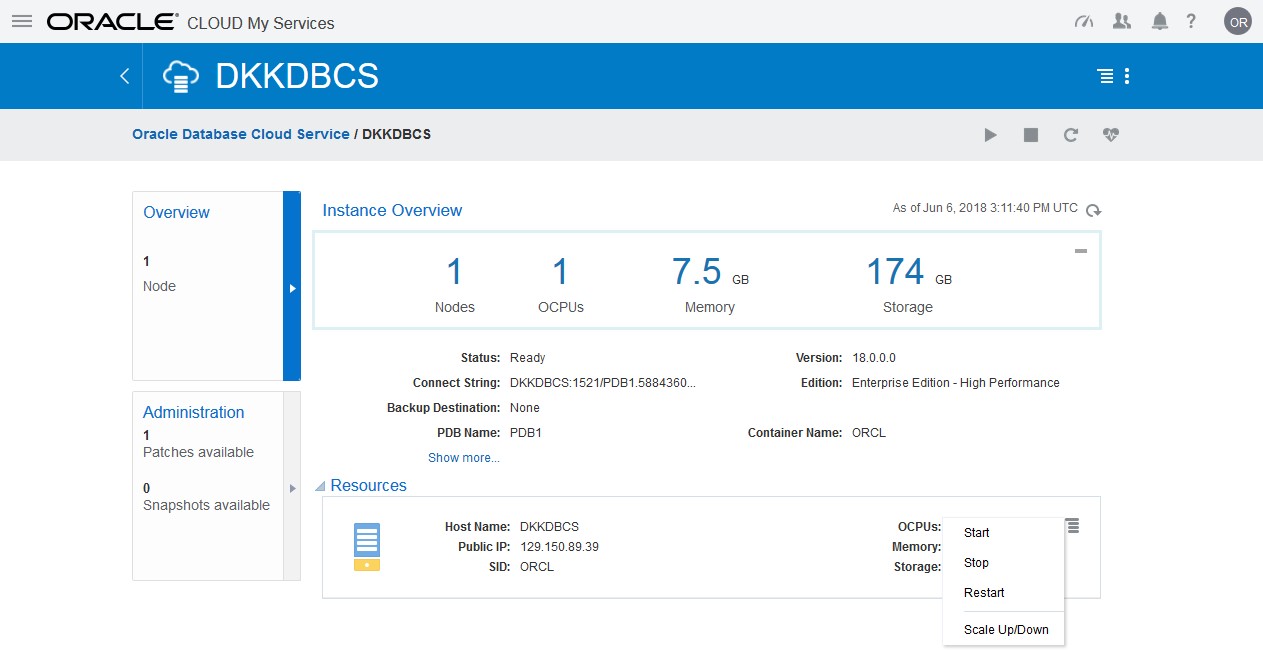
Filesystem Size Used Avail Use% Mounted on

/dev/mapper/redoVolGroup-lvol0

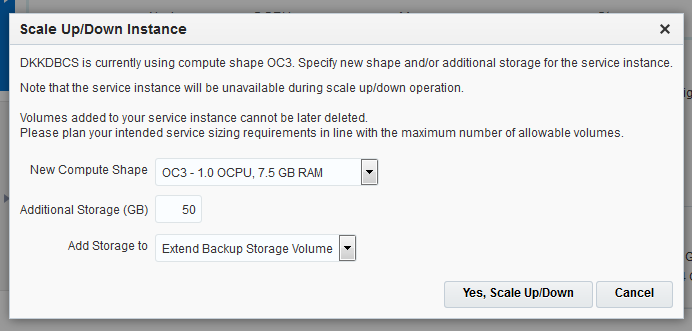
25G 3.3G 20G 14% /u04

[oracle@MYDBCS ~]$

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| /dev/mapper/vg\_main-lv\_root | | | | | |
|  | 25G | 3.7G | 21G | 16% | / |
| tmpfs | 3.7G | 0 | 3.7G | 0% | /dev/shm |
| /dev/xvdb1 | 477M | 70M | 379M | 16% | /boot |
| /dev/xvde1 | 59G | 9.7G | 47G | 18% | /u01 |
| /dev/mapper/dataVolGroup-lvol0 | | | | | |
| 50G | | 7.8G | 39G | 17% | /u02 |
| /dev/mapper/fraVolGroup-lvol0 | | | | | |
| 6.8G | | 5.6G | 854M | 87% | /u03 |

1. If the used storage (Use%) on /u03 is more than 50%, add storage to the database deployment.
   1. Close the connection to the compute node.
   2. Open the browser and access the Oracle Cloud web site by entering https://cloud.oracle.com/home.
   3. Click **Sign In**.
   4. On the Cloud Account page, select **Cloud Account with Identity Cloud Service** in the menu and enter the account name as provided. Click **My Services**.
   5. On the Oracle Cloud Account Sign In page, enter the user name and password as provided. Click **Sign In**.
   6. Expand the menu next to Oracle Cloud My Services.
   7. Expand the **Services** menu and click **Database**.
   8. Search for your database deployment (instance) by entering its name in the search field.
   9. Click the link on the database deployment (instance) name.
   10. Expand the menu in the Resources area and select **Scale Up/Down**.

nly.

* 1. In the Scale Up/Down Instance window, enter **50** in the “Additional Storage (GB)” field and select **Extend Backup Storage Volume** in the “Add Storage to” menu.
  2. Click **Yes, Scale Up/Down**.
  3. A confirmation message is displayed. Periodically refresh the page until you see that the status is once again Ready.
  4. Sign out of your Cloud account.

1. Return to your terminal window and connect to the compute node.

oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. In your terminal window, check the use of disk storage again to ensure you now have room for backups.

[oracle@MYDBCS ~]$ **df -h**

Filesystem Size Used Avail Use% Mounted on

56G 6.3G 47G 12% /u03

[oracle@MYDBCS ~]$

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| /dev/mapper/vg\_main-lv\_root | | | | | |
|  | 25G | 3.7G | 21G | 16% | / |
| tmpfs | 3.7G | 0 | 3.7G | 0% | /dev/shm |
| /dev/xvdb1 | 477M | 70M | 379M | 16% | /boot |
| /dev/xvde1 | 59G | 9.7G | 47G | 18% | /u01 |
| /dev/mapper/dataVolGroup-lvol0 | | | | | |
| 50G | | 7.8G | 39G | 17% | /u02 |
| /dev/mapper/redoVolGroup-lvol0 | | | | | |
| 25G | | 3.3G | 20G | 14% | /u04 |
| /dev/mapper/fraVolGroup-lvol0 | | | | | |

**Practice 18-4: Creating a Whole Database Backup**

### Overview

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

### Assumptions

You are logged in to the compute node as the oracle user. You completed the following practices:

* Practice 17-2 Configure the Size of the Fast Recovery Area
* Practice 18-2 Verifying Automatic Backups of the Control File and SPFILE
* Practice 18-5: Checking Storage Availability

### Tasks

1. Start Oracle Recovery Manager (RMAN) and connect to the CDB root as the SYS user.

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

Recovery Manager: Release 18.0.0.0.0 - Production on Thu Apr 5 13:43:04 2018

Version 18.1.0.0.0

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

connected to target database: ORCL (DBID=2299035716) RMAN>

1. View the structure of the CDB in terms of PDBs, tablespaces, and data files (permanent and temporary). Your file numbers will differ from those shown below.

---- -------- -------------------- ------- ---------------------

1 870 SYSTEM YES

/u02/app/oracle/oradata/ORCL/system01.dbf

RB segs Datafile Name

List of Permanent Datafiles

===========================

File Size(MB) Tablespace

RMAN> **REPORT schema;**

using target database control file instead of recovery catalog Report of database schema for database with db\_unique\_name ORCL

1. 1480 SYSAUX NO

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

1. 60 UNDOTBS1 YES

/u02/app/oracle/oradata/ORCL/undotbs01.dbf

1. 340 PDB$SEED:SYSTEM NO

/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf

1. 620 PDB$SEED:SYSAUX NO

/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf

1. 5 USERS NO

/u02/app/oracle/oradata/ORCL/users01.dbf

1. 200 PDB$SEED:UNDOTBS1 NO

/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf

1. 350 PDB1:SYSTEM YES

/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf

1. 790 PDB1:SYSAUX NO

/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf

1. 200 PDB1:UNDOTBS1 YES

/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf

1. 50 PDB1:USERS NO

/u02/app/oracle/oradata/ORCL/PDB1/PDB1\_users01.dbf

1. 350 PDB2:SYSTEM YES

/u02/app/oracle/oradata/ORCL/PDB2/system01.dbf

1. 670 PDB2:SYSAUX NO

/u02/app/oracle/oradata/ORCL/PDB2/sysaux01.dbf

1. 200 PDB2:UNDOTBS1 YES

/u02/app/oracle/oradata/ORCL/PDB2/undotbs01.dbf

38 7 PDB1:INVENTORY NO

/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF

List of Temporary Files

=======================

File Size(MB) Tablespace Maxsize(MB) Tempfile Name

---- -------- -------------------- ----------- -----------------

1 130 TEMP 32767

/u04/app/oracle/oradata/temp/temp01.dbf

2 62 PDB$SEED:TEMP 32767

/u04/app/oracle/oradata/temp/pdbseed\_temp012018-02-08\_13-49-27- 256-PM.dbf

3 62 PDB2:TEMP 32767

/u04/app/oracle/oradata/temp/PDB2/pdbseed\_temp012018-02-08\_13- 49-27-256-PM.dbf

4 62 PDB1:TEMP 32767

/u04/app/oracle/oradata/temp/temp012018-02-08\_13-49-27-256- PM.dbf

RMAN>

1. Back up the whole database. Your results will be different from the results shown below; for example, the piece handle names will be different.

RMAN> **BACKUP DATABASE;**

Starting backup at 09-APR-18

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=46 device type=DISK

channel ORA\_DISK\_1: starting full datafile backup set

channel ORA\_DISK\_1: specifying datafile(s) in backup set input datafile file number=00001 name=/u02/app/oracle/oradata/ORCL/system01.dbf

input datafile file number=00003 name=/u02/app/oracle/oradata/ORCL/sysaux01.dbf

input datafile file number=00004 name=/u02/app/oracle/oradata/ORCL/undotbs01.dbf

input datafile file number=00007 name=/u02/app/oracle/oradata/ORCL/users01.dbf

channel ORA\_DISK\_1: starting piece 1 at 09-APR-18 channel ORA\_DISK\_1: finished piece 1 at 09-APR-18 piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/backupset/2018\_04

\_09/o1\_mf\_nnndf\_TAG20180409T160501\_fdqkvh9t\_.bkp tag=TAG20180409T160501 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:01:25 channel ORA\_DISK\_1: starting full datafile backup set

channel ORA\_DISK\_1: specifying datafile(s) in backup set

input datafile file number=00013 name=/u02/app/oracle/oradata/ORCL/PDB1/sysaux01

…

channel ORA\_DISK\_1: finished piece 1 at 09-APR-18

piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/69350B8874FA03C8E 053A23F160AC9F7/backupset/2018\_04\_09/o1\_mf\_nnndf\_TAG20180409T160 501\_fdql1xjr\_.bkp tag=TAG20180409T160501 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:00:55 Finished backup at 09-APR-18

Starting Control File and SPFILE Autobackup at 09-APR-18 piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0 4\_09/o1\_mf\_s\_973008564\_fdql3o2s\_.bkp comment=NONE

Finished Control File and SPFILE Autobackup at 09-APR-18 RMAN>

1. Question: Do you have to shut down the database to back it up?

Answer: No, as long as the database is in ARCHIVELOG mode, the backup can take place while the database is opened. This is a hot backup (or online backup). A cold backup (or offline backup) is a backup completed while the database is closed and is required if the database is in NOARCHIVELOG mode.

1. Question: Are hot backups consistent?

Answer: Online backups are inconsistent because with the database opened, there is no guarantee that the data files are synchronized with the control files. However, offline backups taken while the database is not opened are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

1. Question: What would allow hot backups (inconsistent backups) to perform a complete database recovery?

Answer: During a complete recovery, restored online backups are recovered until the current SCN is matched, with the use of the archive log files and online redo log files.

1. Question: Did the backup include the SPFILE and control files?

Answer: Yes. This is the last operation completed at the end of the backup command.

…

Starting Control File and SPFILE Autobackup at 09-APR-18 piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0

4\_09/o1\_mf\_s\_973008564\_fdql3o2s\_.bkp comment=NONE

Finished Control File and SPFILE Autobackup at 09-APR-18

1. Question: Does the complete operation create a single backup set? Answer: No. The operation creates multiple backup sets.
   * Four backup sets including data files (one for each of the containers): CDB root, PDB seed, PDB1, PDB2
   * One backup set for the SPFILE and control files.
2. List the backup sets. Look for Piece Name in the results for each backup set.

Compressed: NO Tag:

Status: AVAILABLE

BP Key: 2 TAG20180405T170539

------- ---- -- ---------- ----------- ------------ ------------

---

2 Full 17.95M DISK 00:00:01 05-APR-18

Device Type Elapsed Time Completion

Type LV Size

BS Key Time

RMAN> **LIST BACKUP;**

List of Backup Sets

===================

Piece Name:

/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_04\_05/o1

\_mf\_s\_972666339\_fdf3x4bf\_.bkp

SPFILE Included: Modification time: 05-APR-18 SPFILE db\_unique\_name: ORCL

Control File Included: Ckp SCN: 2845204 Ckp time: 05-APR- 18

…

BS Key Type LV Size Device Type Elapsed Time Completion Time

------- ---- -- ---------- ----------- ------------ ------------

---

10 Full 18.23M DISK 00:00:01 09-APR-18

BP Key: 10 Status: AVAILABLE Compressed: NO Tag: TAG20180409T160924

Piece Name:

/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_04\_09/o1

\_mf\_s\_973008564\_fdql3o2s\_.bkp

SPFILE Included: Modification time: 09-APR-18 SPFILE db\_unique\_name: ORCL

Control File Included: Ckp SCN: 3108607 Ckp time: 09-APR- 18

RMAN>

1. Exit RMAN.

RMAN> **EXIT**

1. Verify that the files are stored on disk in the FRA.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| $ **cd /u03/app/oracle/fast\_recovery\_area/ORCL**  $ **ls –ltR**  .:  total 18608  -rwxr-x--- 1 oracle oinstall 19021824 Apr 9 16:19 control02.ctl  drwxr-x--- 3 oracle oinstall 4096 Apr 9 16:08 69350B8874FA03C8E053A23F160AC9F7  drwxr-x--- 3 oracle oinstall 4096 Apr 9 16:07 64B4AE270F061AFDE0536604C40A9006  drwxr-x--- 3 oracle oinstall 4096 Apr 9 16:06 692121551ED82717E053E20D130AEB6C | | | | | | | | |
| drwxr-x--- | 3 | oracle | oinstall | 4096 | Apr | 9 | 16:05 | backupset |
| drwxr-x--- | 5 | oracle | oinstall | 4096 | Apr | 9 | 15:01 | autobackup |
| drwxr-x--- | 5 | oracle | oinstall | 4096 | Apr | 9 | 14:33 | archivelog |
| drwxr-x--- | 2 | oracle | oinstall | 4096 | Apr | 5 | 16:51 | flashback |
| drwxr-x--- | 2 | oracle | oinstall | 4096 | Apr | 5 | 16:49 | onlinelog |

./onlinelog: total 0

[oracle@MYDBCS ORCL]$

5 16:52

9 16:15

./flashback:

total 2097176

-rwxr-x--- 1 oracle oinstall 1073750016 Apr o1\_mf\_fdf30qbz\_.flb

-rwxr-x--- 1 oracle oinstall 1073750016 Apr o1\_mf\_fdf32pm6\_.flb

./69350B8874FA03C8E053A23F160AC9F7:

total 4

drwxr-x--- 3 oracle oinstall 4096 Apr 9 16:08 backupset

…

./archivelog/2018\_04\_06: total 0

./archivelog/2018\_04\_05: total 0

1. Question: Where are the backups of control files and SPFILE located? Answer: They are created in the autobackup subdirectory.
2. Question: How are backups deleted?

Answer: Space management in the FRA is governed by a backup retention policy. A retention policy determines when files are obsolete, which means that they are no longer needed to meet your data recovery objectives. The Oracle Database server automatically manages this storage by deleting files that are no longer needed.

1. View the backup retention policy.
2. Start RMAN and connect to the CDB root as the SYS user.

$ **rman target /**

1. Issue the SHOW RETENTION POLICY command. The policy is REDUNDANCY 1.

RMAN> **SHOW RETENTION POLICY;**

using target database control file instead of recovery catalog

RMAN configuration parameters for database with db\_unique\_name ORCL are:

CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default

RMAN>

1. Question: How does Oracle determine when files are obsolete?

Answer: There are two retention policy parameters that are mutually exclusive:

* + If a retention policy is enabled with RECOVERY WINDOW OF 5 DAYS, the window stretches from the current time (SYSDATE) to the point of recoverability, which is the earliest date to which you want to recover. The point of recoverability is SYSDATE - integer days in the past.
  + If a retention policy is enabled with REDUNDANCY r, then RMAN skips backups only if at least *n* backups of an identical file exist on the specified device, where n=r+1 (default is 1).

RMAN automatically deletes obsolete backup sets and copies in the FRA when space is needed.

1. Manually delete obsolete files by issuing the DELETE OBSOLETE command.

RMAN> **delete obsolete;**

using target database control file instead of recovery catalog RMAN retention policy will be applied to the command

RMAN retention policy is set to redundancy 1 allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=4 device type=DISK Deleting the following obsolete backups and copies:

Type Key Completion Time Filename/Handle

-------------------- ------ ------------------ -----------------

---

Backup Set 2 05-APR-18

Backup Piece 2 05-APR-18

/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_04\_05/o1

\_mf\_s\_972666339\_fdf3x4bf\_.bkp

|  |  |  |
| --- | --- | --- |
| Backup Set | 3 | 06-APR-18 |
| Backup Piece | 3 | 06-APR-18 |

/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_04\_06/o1

\_mf\_s\_972751516\_fdhq2y5d\_.bkp

|  |  |  |
| --- | --- | --- |
| Backup Set | 4 | 09-APR-18 |
| Backup Piece | 4 | 09-APR-18 |

/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_04\_09/o1

\_mf\_s\_973004494\_fdqg4gqp\_.bkp

|  |  |  |
| --- | --- | --- |
| Backup Set | 5 | 09-APR-18 |
| Backup Piece | 5 | 09-APR-18 |

/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_04\_09/o1

\_mf\_s\_973005697\_fdqhb2cd\_.bkp

Do you really want to delete the above objects (enter YES or NO)? **yes**

deleted backup piece

backup piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0 4\_05/o1\_mf\_s\_972666339\_fdf3x4bf\_.bkp RECID=2 STAMP=972666340

deleted backup piece

backup piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0 4\_06/o1\_mf\_s\_972751516\_fdhq2y5d\_.bkp RECID=3 STAMP=972751517

deleted backup piece backup piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0

4\_09/o1\_mf\_s\_973004494\_fdqg4gqp\_.bkp RECID=4 STAMP=973004494 deleted backup piece

backup piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0 4\_09/o1\_mf\_s\_973005697\_fdqhb2cd\_.bkp RECID=5 STAMP=973005698

Deleted 4 objects

RMAN>

1. Back up the database and archive logs as image copies. At the same time, free space in the FRA by deleting the archive log files once they are backed up.
2. Perform the backup.

RMAN> **BACKUP AS COPY DATABASE PLUS ARCHIVELOG DELETE INPUT;**

Starting backup at 06-JUN-18 current log archived

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=284 device type=DISK

channel ORA\_DISK\_1: starting archived log copy

input archived log thread=1 sequence=15 RECID=15 STAMP=978029619 output file name=/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_06\_

06/o1\_mf\_1\_15\_fkj1fpws\_.arc RECID=21 STAMP=978105797

…

Finished backup at 06-JUN-18

Starting backup at 06-JUN-18 using channel ORA\_DISK\_1

channel ORA\_DISK\_1: starting datafile copy

input datafile file number=00003 name=/u02/app/oracle/oradata/ORCL/sysaux01.dbf

…

Starting Control File and SPFILE Autobackup at 06-JUN-18

piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0 6\_06/o1\_mf\_s\_978106455\_fkj22r3p\_.bkp comment=NONE

Finished Control File and SPFILE Autobackup at 06-JUN-18

RMAN>

1. Question: What would you do if an error such as the following occurs?

RMAN-00571:

===========================================================

RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS

=============== RMAN-00571:

===========================================================

RMAN-03002: failure of backup plus archivelog command at 01/23/2017 11:05:08

ORA-19809: limit exceeded for recovery files

ORA-19804: cannot reclaim 67108864 bytes disk space from 19327352832 bytes limit

Answer: Increase the DB\_RECOVERY\_FILE\_DEST\_SIZE parameter value to 30G by issuing the following command:

RMAN> ALTER SYSTEM SET db\_recovery\_file\_dest\_size = 30G SCOPE=both;

1. Question: What is the advantage of creating backups as image copies?

Answer: The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy, only the file or files need to be retrieved from your backup location. With backup sets, the entire backup set must be retrieved from your backup location before you extract the file or files that are needed.

1. Question: What is the advantage of creating backups as backup sets?

Answer: The advantage of creating backups as backup sets is better space usage. In most databases, 20% or more of the data blocks are empty blocks. Image copies back up every data block, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems, the advantages of backup sets outweigh the advantages of image copies.

1. Question: How many image copies of the data files are created?

Answer: There are 15 image copies, one image copy for each data file in the CDB, PDBs included.

1. Exit RMAN.

RMAN> **EXIT**

**Practice 18-6: Creating Partial Database Backups**

### Overview

In this practice, you use Recovery Manager to back up PDB1, including the archived redo log files. This is a partial database backup.

### Assumptions

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

### Tasks

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

[oracle@MYDBCS ORCL]$ **rman target /**

Recovery Manager: Release 18.0.0.0.0 - Production on Wed Jun 6 16:48:45 2018

Version 18.1.0.0.0

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

connected to target database: ORCL (DBID=1505229725) RMAN>

1. Back up PDB1, including the archived redo log files.

RMAN> **BACKUP PLUGGABLE DATABASE PDB1 PLUS ARCHIVELOG;**

Starting backup at 06-JUN-18 current log archived

using target database control file instead of recovery catalog

allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=41 device type=DISK

channel ORA\_DISK\_1: starting archived log backup set channel ORA\_DISK\_1: specifying archived log(s) in backup set

input archived log thread=1 sequence=14 RECID=22 STAMP=978105812

…

Starting Control File and SPFILE Autobackup at 06-JUN-18 piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0

6\_06/o1\_mf\_s\_978108946\_fkj4jm29\_.bkp comment=NONE Finished Control File and SPFILE Autobackup at 06-JUN-18

RMAN>

1. Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete.

[oracle@MYDBCS ORCL]$

1. Question: Did the partial backup automatically include the SPFILE and control files?

Answer: Yes. The setting verified in Practice 18-2 Verifying Automatic Backups of the Control File and SPFILE is also valid for partial backups.

1. Question: How many backup sets are created?

Answer: Four backup sets: one for the PDB data files, one for the SPFILE and control file, one for the archived log files before the data file backup set, and one for the archived log files after the data file backup set.

1. Question: Can you connect in RMAN directly to the PDB to perform the same backup?

Answer: Yes. In this case, you do not have to specify that you want to back up a PDB. Instead, you can use the simple BACKUP DATABASE command.

1. Perform a partial database backup in PDB1 directly.
   1. Start RMAN and connect to PDB1 as the SYS user.

[oracle@MYDBCS ORCL]$ **rman target SYS/*password*@PDB1**

**…**

connected to target database: ORCL:PDB1 (DBID=2133829969)

RMAN>

* 1. Execute the BACKUP DATABASE command. Notice that the SPFILE and control file are not backed up.

RMAN> **BACKUP DATABASE;**

Starting backup at 06-JUN-18

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=41 device type=DISK

…

channel ORA\_DISK\_1: starting piece 1 at 06-JUN-18 channel ORA\_DISK\_1: finished piece 1 at 06-JUN-18

piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/6D5DA61701C1285EE 0533E89810ACAA5/backupset/2018\_06\_06/o1\_mf\_nnndf\_TAG20180606T165 917\_fkj4q7w6\_.bkp tag=TAG20180606T165917 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:01:25 Finished backup at 06-JUN-18

RMAN>

1. Try to configure the recovery setting for the PDB so that the SPFILE and control file are backed up too. You get an error message because you must be connected to the CDB root to configure any recovery settings.

RMAN> **CONFIGURE CONTROLFILE AUTOBACKUP ON;**

RMAN-00571:

=========================================================== RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS

===============

RMAN-00571:

===========================================================

RMAN-03002: failure of configure command at 06/06/2018 17:02:27

RMAN-07536: command not allowed when connected to a Pluggable Database

RMAN>

1. Exit RMAN.

RMAN> **EXIT**

1. Back up the TBS\_APP tablespace in PDB2.
   1. Connect to PDB2 as the SYS user.

$ **rman target SYS/*password*@PDB2**

…

connected to target database: ORCL:PDB2 (DBID=3237529478)

RMAN>

* 1. Back up the tablespace.

RMAN> **BACKUP TABLESPACE tbs\_app;**

Starting backup at 06-JUN-18

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=55 device type=DISK

channel ORA\_DISK\_1: starting full datafile backup set

channel ORA\_DISK\_1: specifying datafile(s) in backup set input datafile file number=00059 name=/u02/app/oracle/oradata/ORCL/PDB2/tbs\_app01.dbf

channel ORA\_DISK\_1: starting piece 1 at 06-JUN-18 channel ORA\_DISK\_1: finished piece 1 at 06-JUN-18

piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/6D975E8B80B85F14E 0537A051D0A3C0D/backupset/2018\_06\_06/o1\_mf\_nnndf\_TAG20180606T170 407\_fkj5091o\_.bkp tag=TAG20180606T170407 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:00:04 Finished backup at 06-JUN-18

RMAN>

* 1. Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete.

[oracle@DKKDBCS ORCL]$

1. Can you connect to the CDB root and perform the same operation?
   1. Start RMAN and connect to the CDB root as the SYS user.

$ **rman target /**

…

connected to target database: ORCL (DBID=1505229725)

RMAN>

* 1. Back up the TBS\_APP tablespace in PDB2. You must specify the PDB in which the tablespace exists.

##### RMAN> BACKUP TABLESPACE PDB2:tbs\_app;

Starting backup at 06-JUN-18

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=271 device type=DISK

channel ORA\_DISK\_1: starting full datafile backup set channel ORA\_DISK\_1: specifying datafile(s) in backup set

input datafile file number=00059 name=/u02/app/oracle/oradata/ORCL/PDB2/tbs\_app01.dbf

channel ORA\_DISK\_1: starting piece 1 at 06-JUN-18

channel ORA\_DISK\_1: finished piece 1 at 06-JUN-18 piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/6D975E8B80B85F14E

0537A051D0A3C0D/backupset/2018\_06\_06/o1\_mf\_nnndf\_TAG20180606T170 547\_fkj53f1h\_.bkp tag=TAG20180606T170547 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:00:03 Finished backup at 06-JUN-18

Starting Control File and SPFILE Autobackup at 06-JUN-18 piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0

6\_06/o1\_mf\_s\_978109551\_fkj53l2p\_.bkp comment=NONE Finished Control File and SPFILE Autobackup at 06-JUN-18

RMAN>

* 1. Question: Did the operation back up only the tablespace data files?

Answer: No. It also backed up the SPFILE and control file. It is only when you are connected to the CDB root to perform backups that the SPFILE and control file are backed up.

* 1. Exit RMAN and close the terminal window.

RMAN> **EXIT**

**Practices for Lesson 19: Performing Database**

**Recovery**

**Practices for Lesson 19: Overview**

### Overview

In these practices, you will initiate a recovery operation by using RMAN commands. You will use the Data Recovery Advisor to recover a datafile.

**Practice 19-1: Recovering from the Loss of a System-Critical Data File**

### Overview

In this practice, you recover your CDB after the data file for the SYSTEM tablespace (in the CDB root) has been inadvertently removed.

##### Tip

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

To set a title for a terminal window:

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

### Assumptions

You are logged in as the oracle user.

You completed the following practices:

* + Practice 18-2 Verifying Automatic Backups of the Control File and SPFILE
  + Practice 18-4 Creating a Whole Database Backup

### Tasks

##### Create a Loss of a System-Critical Data File Window 1

1. Open a new terminal window and connect to the compute node. This window will be referred to as Window 1.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Execute the $HOME/labs/RMAN\_crash.sh shell script to remove the data file of the

SYSTEM tablespace in the CDB root.

[oracle@MYDBCS ~]$ **$HOME/labs/RMAN\_crash.sh**

System altered.

System altered.

ALTER DATABASE DATAFILE

'/u02/app/oracle/oradata/ORCL/system01.dbf' RESIZE 5M

\*

ERROR at line 1:

ORA-01116: error in opening database file 1

ORA-01110: data file 1: '/u02/app/oracle/oradata/ORCL/system01.dbf'

ORA-27041: unable to open file

Linux-x86\_64 Error: 2: No such file or directory Additional information: 3

[oracle@MYDBCS ~]$

1. Attempt an administrative task, such as creating a user.
   1. Start SQL\*Plus and connect to the CDB root as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus system/*password***

…

SQL>

* 1. Create a common user named c##test. You get an error telling you that data file #1 is missing, and it is the one associated with the SYSTEM tablespace in the CDB root.

SQL> **CREATE USER c##test IDENTIFIED BY DBAdmin\_1;**

CREATE USER c##test IDENTIFIED BY DBAdmin\_1

\*

ERROR at line 1:

ORA-00604: error occurred at recursive SQL level 1 ORA-01116: error in opening database file 1

ORA-01110: data file 1: '/u02/app/oracle/oradata/ORCL/system01.dbf' ORA-27041: unable to open file

Linux-x86\_64 Error: 2: No such file or directory Additional information: 3

SQL>

* 1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Consider your recovery options.
   1. Question: Which type of recovery is possible in this case?

Answer: A complete recovery is possible as long as you have all available backups required. This means that you have a backup (backup set or image copy) of the missing data file, all archive log files required to recover the restored data file up to the current SCN of the CDB including all redo log files (one member in each group will be sufficient).

* 1. Question: Which methods can you use to recover the data file?
  2. Answer: RMAN is the best utility to recover data. You can use the RESTORE and

RECOVER commands or get help by using the LIST FAILURE command.

##### Recover the Database by Using the RESTORE and RECOVER Commands Window 1

1. Start Recovery Manager (RMAN) and connect to the target database (CDB root).

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

…

connected to target database: ORCL (DBID=1500451933)

RMAN>

1. Issue the RESTORE command. You must provide the number for the missing data file.

RMAN> **RESTORE DATAFILE 1;**

Starting restore at 13-JUN-18

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=46 device type=DISK

channel ORA\_DISK\_1: restoring datafile 00001 input datafile copy RECID=2 STAMP=978678712 file

name=/u03/app/oracle/fast\_recovery\_area/ORCL/datafile/o1\_mf\_syst

em\_fl1jwb73\_.dbf

destination for restore of datafile 00001:

/u02/app/oracle/oradata/ORCL/system01.dbf

RMAN-00571:

===========================================================

RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS

===============

RMAN-00571:

===========================================================

RMAN-03002: failure of restore command at 06/13/2018 08:08:12

ORA-19573: cannot obtain exclusive enqueue for datafile 1

ORA-45909: restore, recover or block media recovery may be in progress

ORA-19600: input file is datafile-copy 2 (/u03/app/oracle/fast\_recovery\_area/ORCL/datafile/o1\_mf\_system\_f l1jwb73\_.dbf)

ORA-19601: output file is datafile 1 (/u02/app/oracle/oradata/ORCL/system01.dbf)

RMAN>

1. Question: What does the error message "cannot obtain exclusive enqueue for datafile 1" mean?

Answer: The restore operation requires an exclusive enqueue lock on the data file 1 that is not obtainable. In this case, you have to open the database instance in MOUNT mode. This means that you have to shut down the database instance if it did not already abort.

1. Question: What does a database instance shut down do?

Answer: This closes all PDBs and therefore prevents all users from working during the recovery operation.

1. Try to shut down the database instance in IMMEDIATE mode. You get an error.

RMAN> **SHUTDOWN IMMEDIATE**

RMAN-00571:

=========================================================== RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS

===============

RMAN-00571:

===========================================================

RMAN-03002: failure of shutdown command at 06/13/2018 08:10:09 ORA-01116: error in opening database file 1

ORA-01110: data file 1: '/u02/app/oracle/oradata/ORCL/system01.dbf' ORA-27041: unable to open file

Linux-x86\_64 Error: 2: No such file or directory Additional information: 3

RMAN>

1. Question: Why does the SHUTDOWN IMMEDIATE command fail?

Answer: RMAN needs to close all data files cleanly by writing the current SCN to all data file headers. This cannot be done because data file 1 is missing.

1. Reconnect to the database.
2. Try to shut down the database instance in ABORT mode.

RMAN> **SHUTDOWN ABORT**

Oracle instance shut down

RMAN>

1. Start the database instance in MOUNT mode.

|  |  |  |
| --- | --- | --- |
| RMAN> **STARTUP MOUNT**  Oracle instance started database mounted |  | |
| Total System Global Area | 2768239832 | bytes |
| Fixed Size | 8899800 | bytes |
| Variable Size | 704643072 | bytes |
| Database Buffers | 1979711488 | bytes |
| Redo Buffers | 74985472 | bytes |
| RMAN> |  |  |
| 10. Restore the missing data file. |  |  |
| RMAN> **RESTORE DATAFILE 1;** |  |  |
| Starting restore at 10-APR-18 allocated channel: ORA\_DISK\_1  channel ORA\_DISK\_1: SID=25 device type=DISK  channel ORA\_DISK\_1: starting datafile backup set restore  channel ORA\_DISK\_1: specifying datafile(s) to restore from backup set  channel ORA\_DISK\_1: restoring datafile 00001 to  /u02/app/oracle/oradata/ORCL/system01.dbf channel ORA\_DISK\_1: reading from backup piece  /u03/app/oracle/fast\_recovery\_area/ORCL/backupset/2018\_04\_09/o1\_  mf\_nnndf\_TAG20180409T160501\_fdqkvh9t\_.bkp  channel ORA\_DISK\_1: piece handle=/u03/app/oracle/fast\_recovery\_area/ORCL/backupset/2018\_04  \_09/o1\_mf\_nnndf\_TAG20180409T160501\_fdqkvh9t\_.bkp tag=TAG20180409T160501  channel ORA\_DISK\_1: restored backup piece 1  channel ORA\_DISK\_1: restore complete, elapsed time: 00:00:15 Finished restore at 10-APR-18  RMAN> | | |

1. Recover the missing data file.

RMAN> **RECOVER DATAFILE 1;**

Starting recover at 10-APR-18 using channel ORA\_DISK\_1

starting media recovery

archived log for thread 1 with sequence 14 is already on disk as file

/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_10/o1

\_mf\_1\_14\_fdskgq34\_.arc

archived log for thread 1 with sequence 15 is already on disk as file

/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_10/o1

\_mf\_1\_15\_fdsmhyob\_.arc

archived log for thread 1 with sequence 16 is already on disk as file

/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_10/o1

\_mf\_1\_16\_fdsmhyqz\_.arc

archived log file name=/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_ 10/o1\_mf\_1\_14\_fdskgq34\_.arc thread=1 sequence=14

media recovery complete, elapsed time: 00:00:01 Finished recover at 10-APR-18

RMAN>

1. Open the CDB root.

RMAN> **ALTER DATABASE OPEN;**

Statement processed RMAN>

1. Open all PDBs.

RMAN> **ALTER PLUGGABLE DATABASE ALL OPEN;**

Statement processed RMAN>

1. Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete. [oracle@MYDBCS ~]$

1. Start SQL\*Plus and connect to the CDB root as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **sqlplus SYSTEM/*password***

…

SQL>

1. Try creating the c##test user again. This time the user is created.

SQL> **CREATE USER c##test IDENTIFIED BY DBAdmin\_1;**

User created. SQL>

1. Keep Window 1 open for the next section.

##### Use the Data Recovery Advisor to Recover the Database

1. **Window 2:** Open a new terminal window and execute the RMAN\_crash.sh script to create a failure. This window will be referred to as Window 2.

$ **$HOME/labs/RMAN\_crash.sh**

System altered.

System altered.

ALTER DATABASE DATAFILE

'/u02/app/oracle/oradata/ORCL/system01.dbf' RESIZE 5M

\*

ERROR at line 1:

ORA-01116: error in opening database file 1 ORA-01110: data file 1: '/u02/app/oracle/oradata/ORCL/system01.dbf'

ORA-27041: unable to open file

Linux-x86\_64 Error: 2: No such file or directory Additional information: 3

[oracle@MYDBCS ~]$

1. **Window 1:** Try to create another user named c##test2. The user is successfully created.

SQL> **CREATE USER c##test2 IDENTIFIED BY *password*;**

User created. SQL>

1. Question: How is it possible that you were able to create the user immediately following the crash scenario?

Answer: Remember that the DBWR background process does not necessarily write immediately into the data files.

1. **Window 1:** Attempt to resize datafile 1. If it completes, execute the ALTER SYSTEM SWITCH LOGFILE command. You should receive an error message about the missing data file.

SQL> **ALTER DATABASE DATAFILE 1 RESIZE 1G;**

ALTER DATABASE DATAFILE 1 RESIZE 1G

\*

ERROR at line 1:

ORA-01565: error in identifying file '/u02/app/oracle/oradata/ORCL/system01.dbf' ORA-27037: unable to obtain file status

Linux-x86\_64 Error: 2: No such file or directory Additional information: 7

SQL>

1. **Window 1:** Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. **Window 1:** Start RMAN and connect to the target database.

$ **rman target /**

…

connected to target database (not started)

RMAN>

1. **Window 1:** Start the database instance in MOUNT mode.

RMAN>

8899800 bytes

704643072 bytes

1979711488 bytes

74985472 bytes

Fixed Size Variable Size Database Buffers

Redo Buffers

2768239832 bytes

RMAN> **STARTUP MOUNT;**

Oracle instance started database mounted

Total System Global Area

1. **Window 1:** Use the LIST FAILURE command to determine the error. The value in the

Summary column tells you that system01.dbf is missing.

---------- -------- --------- ------------- -------

262 CRITICAL OPEN 10-APR-18 System datafile 1: '/u02/app/oracle/oradata/ORCL/system01.dbf' is missing

RMAN>

Time Detected Summary

Failure ID Priority Status

RMAN> **LIST FAILURE;**

using target database control file instead of recovery catalog Database Role: PRIMARY

List of Database Failures

=========================

1. **Window 1:** Display repair options. At the very end of the results, a repair script is listed.

RMAN> **ADVISE FAILURE;**

Database Role: PRIMARY

List of Database Failures

=========================

Failure ID Priority Status Time Detected Summary

---------- -------- --------- ------------- -------

262 CRITICAL OPEN 10-APR-18 System datafile 1: '/u02/app/oracle/oradata/ORCL/system01.dbf' is missing

analyzing automatic repair options; this may take some time allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=25 device type=DISK analyzing automatic repair options complete

Mandatory Manual Actions

========================

no manual actions available

Optional Manual Actions

=======================

1. If file /u02/app/oracle/oradata/ORCL/system01.dbf was unintentionally renamed or moved, restore it

Automated Repair Options

========================

Option Repair Description

------ ------------------

1 Restore and recover datafile 1

Strategy: The repair includes complete media recovery with no data loss

Repair script:

/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco\_2771153169.hm

RMAN>

1. **Window 1:** Use the REPAIR FAILURE PREVIEW command to generate a script with all repair actions and comments.

RMAN> **REPAIR FAILURE PREVIEW;**

Strategy: The repair includes complete media recovery with no data loss

Repair script:

/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco\_2771153169.hm

contents of repair script:

# restore and recover datafile restore ( datafile 1 ); recover datafile 1;

sql 'alter database datafile 1 online';

RMAN>

1. **Window 1:** Use the REPAIR FAILURE command to repair database failures identified by the Data Recovery Advisor. When prompted, enter YES to execute the repair. When prompted to open the database, enter YES.

RMAN> **REPAIR FAILURE;**

Strategy: The repair includes complete media recovery with no data loss

Repair script:

/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco\_2771153169.hm

contents of repair script:

# restore and recover datafile restore ( datafile 1 ); recover datafile 1;

sql 'alter database datafile 1 online';

Do you really want to execute the above repair (enter YES or NO)? **YES**

executing repair script

Starting restore at 10-APR-18 using channel ORA\_DISK\_1

channel ORA\_DISK\_1: starting datafile backup set restore

channel ORA\_DISK\_1: specifying datafile(s) to restore from backup set

channel ORA\_DISK\_1: restoring datafile 00001 to

/u02/app/oracle/oradata/ORCL/system01.dbf channel ORA\_DISK\_1: reading from backup piece

/u03/app/oracle/fast\_recovery\_area/ORCL/backupset/2018\_04\_09/o1\_

mf\_nnndf\_TAG20180409T160501\_fdqkvh9t\_.bkp channel ORA\_DISK\_1: piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/backupset/2018\_04

\_09/o1\_mf\_nnndf\_TAG20180409T160501\_fdqkvh9t\_.bkp tag=TAG20180409T160501

channel ORA\_DISK\_1: restored backup piece 1

channel ORA\_DISK\_1: restore complete, elapsed time: 00:00:07 Finished restore at 10-APR-18

Starting recover at 10-APR-18 using channel ORA\_DISK\_1

starting media recovery

archived log for thread 1 with sequence 14 is already on disk as file

/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_10/o1

\_mf\_1\_14\_fdskgq34\_.arc

archived log for thread 1 with sequence 15 is already on disk as file

…

archived log file name=/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_ 10/o1\_mf\_1\_16\_fdsmhyqz\_.arc thread=1 sequence=16

archived log file name=/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_ 10/o1\_mf\_1\_17\_fdsn3sf0\_.arc thread=1 sequence=17

media recovery complete, elapsed time: 00:00:01 Finished recover at 10-APR-18

sql statement: alter database datafile 1 online

repair failure complete

Do you want to open the database (enter YES or NO)? **YES**

database opened

RMAN>

1. **Window 1:** Open all the PDBs.

RMAN> **ALTER PLUGGABLE DATABASE ALL OPEN;**

Statement processed RMAN>

1. **Window 1:** Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete. [oracle@MYDBCS ~]$

1. **Window 2:** Close the terminal window.

**Practice 19-2: Recovering from the Loss of an Application Data File**

### Overview

In this practice, you recover a PDB data file that has been inadvertently removed.

### Assumptions

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

### Tasks

##### Set Up Your Environment for the Practice

1. Execute the $HOME/labs/setup\_pdb1.sh shell script. This script creates the TBS\_APP

tablespace and OE schema in PDB1. You can ignore object creation error messages.

[oracle@MYDBCS ~]$ **$HOME/labs/setup\_pdb1.sh**

…

1 row created.

1 row created.

Commit complete. [oracle@MYDBCS ~]$

1. Start RMAN and connect to the CDB root as the SYS user.

$ **rman target /**

…

connected to target database: ORCL (DBID=1500451933)

RMAN>

1. Back up PDB1.

RMAN> **BACKUP PLUGGABLE DATABASE pdb1;**

Starting backup at 11-APR-18

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=274 device type=DISK

channel ORA\_DISK\_1: starting full datafile backup set

channel ORA\_DISK\_1: specifying datafile(s) in backup set

input datafile file number=00019 name=/u02/app/oracle/oradata/ORCL/PDB1/tbs\_app01.dbf

… piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/692121551ED82717E

053E20D130AEB6C/backupset/2018\_04\_11/o1\_mf\_nnndf\_TAG20180411T130 648\_fdwj59r0\_.bkp tag=TAG20180411T130648 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:00:55 Finished backup at 11-APR-18

Starting Control File and SPFILE Autobackup at 11-APR-18 piece

handle=/u03/app/oracle/fast\_recovery\_area/ORCL/autobackup/2018\_0 4\_11/o1\_mf\_s\_973170464\_fdwj71bd\_.bkp comment=NONE

Finished Control File and SPFILE Autobackup at 11-APR-18

RMAN>

1. Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete. [oracle@MYDBCS ~]$

##### Remove a Data File

In this section, you run a script that removes a data file from PDB1. You research the problem and discover which data file is missing.

1. Execute the $HOME/labs/RMAN\_crash\_app.sh script.

$ **$HOME/labs/RMAN\_crash\_app.sh**

System altered.

System altered.

[oracle@MYDBCS ~]$

1. Create an application table and insert data into it.
   1. Connect to the CDB root as the SYS user with the SYSDBA privilege.

$ **sqlplus / AS SYSDBA**

… SQL>

* 1. Connect to PDB1 as the SYS user.

SQL> **ALTER SESSION SET CONTAINER = pdb1;**

Session altered.

SQL>

* 1. Create a table named OE.TEST.

SQL> **CREATE TABLE oe.test (c NUMBER);**

Table created.

SQL>

* 1. Try to insert rows into the table. You get an error indicating that the data file named tbs\_app01.dbf is missing in PDB1. Note that in this example the data file number for tbs\_app01.dbf is 19. Identify and remember the file number assigned to tbs\_app01.dbf on your own system. Your number will probably differ from this example.

SQL> **INSERT INTO oe.test VALUES (1);**

INSERT INTO oe.test VALUES (1)

\* ERROR at line 1:

ORA-01116: error in opening database file 19

ORA-01110: data file 19: '/u02/app/oracle/oradata/ORCL/PDB1/tbs\_app01.dbf' ORA-27041: unable to open file

Linux-x86\_64 Error: 2: No such file or directory Additional information: 3

SQL>

1. Question: Why does the CREATE TABLE statement complete and the INSERT statement fail?

Answer: When you create a table, the segment is not created. The CREATE statement updates a table in the SYSTEM tablespace. Only when the first row is inserted into the table is the segment created in the data file.

1. Find out the tablespace to which tbs\_app01.dbf belongs. In this command, substitute the number assigned to tbs\_app01.dbf on your own system for the example value of 19. The result shows that the tablespace is TBS\_APP.

SQL> **SELECT tablespace\_name from dba\_data\_files WHERE file\_id = 19;**

TABLESPACE\_NAME

------------------------------ TBS\_APP

SQL>

1. Check if the TBS\_APP tablespace is still online. The results indicate that it is online.

SQL>

ONLINE

ONLINE

ONLINE

SYSTEM SYSAUX

…

TBS\_APP

------------------------------ ---------

STATUS

TABLESPACE\_NAME

SQL> **SELECT tablespace\_name, status FROM dba\_tablespaces;**

1. Check if PDB1 is still open. The result indicates that PDB1 is still open.

---------- ------------------------------ ---------- ----------

4 PDB1 READ WRITE NO

SQL>

OPEN MODE RESTRICTED

SQL> **SHOW PDBS**

CON\_ID CON\_NAME

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

##### Restore and Recover PDB1

In this section, you use Recovery Manager to restore and recover PDB1.

1. Question: Which type of recovery is possible in this case?

Answer: A complete recovery is possible as long as you have all available backups required. This means that you have a backup (backup set or image copy) of the missing data file and all archive log files required to recover the restored data file up to the current SCN of the PDB including all redo log files (one member in each group will be sufficient).

1. Question: Which methods can you use to recover?

Answer: RMAN is the best utility to recover data. You can use the RESTORE and RECOVER commands, or you can get help with the LIST FAILURE commands. You can also use the simple REPAIR command.

1. Start RMAN and connect to PDB1 as the SYS user. Refer to *Course Practice Environment: Security Credentials* for the password value.

$ **rman target SYS/*password*@PDB1**

…

connected to target database: ORCL:PDB1 (DBID=687597467, not open)

RMAN>

1. Issue the REPORT SCHEMA command to list the names of the data files (permanent and temporary) and tablespaces for PDB1. In this example, data file number 19 is part of the TBS\_APP tablespace in PDB1 and shows a size of 0MB. Identify the entry for the TBS\_APP tablespace on your system.

RMAN> **REPORT SCHEMA;**

using target database control file instead of recovery catalog Report of database schema for database with db\_unique\_name ORCL

List of Permanent Datafiles

===========================

File Size(MB) Tablespace RB segs Datafile Name

---- -------- -------------------- ------- ---------------------

1. 350 SYSTEM NO

/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf

1. 640 SYSAUX NO

/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf

1. 200 UNDOTBS1 NO

/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf

1. 50 USERS NO

/u02/app/oracle/oradata/ORCL/PDB1/PDB1\_users01.dbf

19 0 TBS\_APP NO

/u02/app/oracle/oradata/ORCL/PDB1/tbs\_app01.dbf

List of Temporary Files

=======================

File Size(MB) Tablespace Maxsize(MB) Tempfile Name

---- -------- -------------------- ----------- ----------------- 4 62 TEMP 32767

/u04/app/oracle/oradata/temp/temp012018-02-08\_13-49-27-256-

PM.dbf

RMAN>

1. You must close PDB1 in IMMEDIATE mode (which puts the PDB into MOUNTED mode) before restoring the PDB; otherwise, you will get the error "cannot obtain exclusive enqueue for datafile...".

RMAN> **SHUTDOWN IMMEDIATE;**

database closed

RMAN>

1. Restore PDB1.

RMAN> **RESTORE DATABASE;**

Starting restore at 11-APR-18 allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=286 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 00012 to

/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf

…

channel ORA\_DISK\_1: restore complete, elapsed time: 00:00:35 Finished restore at 11-APR-18

RMAN>

1. Recover the database. Notice the line "starting media recovery." If you had tried to restore and recover just the data file or just the tablespace, you might have encountered media recovery errors for other data files in the tablespace.

RMAN> **RECOVER DATABASE;**

Starting recover at 11-APR-18 using channel ORA\_DISK\_1

starting media recovery

media recovery complete, elapsed time: 00:00:01 Finished recover at 11-APR-18

RMAN>

1. Start PDB1.

RMAN> **STARTUP**

database opened RMAN>

1. Issue the REPORT SCHEMA command again. Notice that the TBS\_APP tablespace now has a size of 800MB.

RMAN> **REPORT SCHEMA;**

Report of database schema for database with db\_unique\_name ORCL

List of Permanent Datafiles

===========================

File Size(MB) Tablespace RB segs Datafile Name

---- -------- -------------------- ------- ---------------------

12 350 SYSTEM NO

/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf

…

19 800 TBS\_APP NO

/u02/app/oracle/oradata/ORCL/PDB1/tbs\_app01.dbf

List of Temporary Files

=======================

File Size(MB) Tablespace Maxsize(MB) Tempfile Name

---- -------- -------------------- ----------- -----------------

4 62 TEMP 32767

/u04/app/oracle/oradata/temp/temp012018-02-08\_13-49-27-256- PM.dbf

RMAN>

1. Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete. [oracle@MYDBCS ~]$

1. Try again to insert data into the OE.TEST table in PDB1.
   1. Connect to the CDB root as the SYS user with the SYSDBA privilege.

$ **sqlplus / AS SYSDBA**

… SQL>

* 1. Connect to PDB1 as the SYS user.

SQL> **ALTER SESSION SET CONTAINER = pdb1;**

Session altered.

SQL>

* 1. Try to insert rows into the table.

SQL> **INSERT INTO oe.test VALUES (1);**

1 row created.

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

##### Use the REPAIR Command

In this section, you re-create the environment where you have a missing data file. You then use RMAN's REPAIR command to perform all the necessary operations (for example, restore and recovery) to fully recover the data file.

1. Execute the script to remove the data file named tbs\_app01.dbf from PDB1 again.

[oracle@MYDBCS ~]$ **$HOME/labs/RMAN\_crash\_app.sh**

System altered. System altered. [oracle@MYDBCS ~]$

1. Create an application table and try to insert data into it.
   1. Connect to the CDB root as the SYS user with the SYSDBA privilege.

$ **sqlplus / AS SYSDBA**

…

SQL>

* 1. Connect to PDB1 as the SYS user.

SQL> **ALTER SESSION SET CONTAINER = pdb1;**

Session altered.

SQL>

* 1. Show the open mode of PDB1. The result shows that PDB1 is still open.

---------- ------------------------------ ---------- ----------

4 PDB1 READ WRITE NO

SQL>

OPEN MODE RESTRICTED

SQL> **SHOW PDBS**

CON\_ID CON\_NAME

* 1. Create a table named OE.TEST2.

SQL> **CREATE TABLE oe.test2 (c NUMBER);**

Table created.

SQL>

* 1. Try to insert rows into the table. You get the same error that you did before: unable to open file. Again, you are not able to insert data into the OE schema because the data file is missing.

SQL> **INSERT INTO oe.test2 VALUES (1);**

INSERT INTO oe.test2 VALUES (1)

\* ERROR at line 1:

ORA-01116: error in opening database file 19 ORA-01110: data file 19:

'/u02/app/oracle/oradata/ORCL/PDB1/tbs\_app01.dbf'

ORA-27041: unable to open file

Linux-x86\_64 Error: 2: No such file or directory Additional information: 3

SQL>

* 1. Close PDB1 to put the PDB into MOUNTED mode.

SQL> **ALTER PLUGGABLE DATABASE pdb1 CLOSE;**

Pluggable database altered. SQL>

Important: You do not want to end this SQL\*Plus session without closing PDB1 first; otherwise, you may get errors when performing the REPAIR operation in RMAN.

1. Exit SQL\*Plus.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

1. Start RMAN and log in to the target database as the SYS user.

$ **rman target /**

…

connected to target database: ORCL (DBID=1500451933)

RMAN>

1. Execute the REPAIR command. This command restores and recovers the data file.

##### RMAN> REPAIR PLUGGABLE DATABASE pdb1;

Starting restore at 11-APR-18

using target database control file instead of recovery catalog allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=48 device type=DISK

Executing: alter database datafile 12 offline Executing: alter database datafile 13 offline Executing: alter database datafile 14 offline Executing: alter database datafile 15 offline Executing: alter database datafile 19 offline

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 00012 to

/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf

…

Finished restore at 11-APR-18

Starting recover at 11-APR-18 using channel ORA\_DISK\_1

starting media recovery

archived log for thread 1 with sequence 25 is already on disk as file

/u03/app/oracle/fast\_recovery\_area/ORCL/archivelog/2018\_04\_11/o1

\_mf\_1\_25\_fdwk4sgf\_.arc

…

media recovery complete, elapsed time: 00:00:02 Executing: alter database datafile 12 online Executing: alter database datafile 13 online Executing: alter database datafile 14 online Executing: alter database datafile 15 online Executing: alter database datafile 19 online Finished recover at 11-APR-18

RMAN>

1. Open PDB1.

RMAN> **ALTER PLUGGABLE DATABASE pdb1 OPEN;**

Statement processed RMAN>

1. Exit RMAN.

RMAN> **EXIT**

Recovery Manager complete. [oracle@MYDBCS ~]$

1. Try to insert data into the OE.TEST2 table again.
   1. Start SQL\*Plus and connect to PDB1 as the SYSTEM user.

$ **sqlplus SYSTEM/*password*@PDB1**

… SQL>

* 1. Issue the INSERT command. The operation succeeds, which means the REPAIR

command recovered the data file.

SQL> **INSERT INTO oe.test2 VALUES (2);**

1 row created.

SQL>

* 1. Commit the transaction.

SQL> **COMMIT;**

Commit complete.

SQL>

* 1. Exit SQL\*Plus and close the terminal window.

SQL> **EXIT**

…

[oracle@MYDBCS ~]$

**Practices for Lesson 20: Monitoring and Tuning Database Performance**

**Practices for Lesson 20: Overview**

### Overview

In these practices, you will view performance information by using Enterprise Manager Database Express. You will also investigate and resolve a locking issue.

**Practice 20-1: Using Enterprise Manager Database Express to Manage Performance**

### Overview

In this practice, you view the performance of the database instance by using Enterprise Manager Database Express (EM Express).

You could use V$ views to analyze performance statistics and metrics, but it is much easier to use EM Express or EM Cloud Control. Whichever tool you use, the key to identifying instance performance issues are wait events and high-cost SQL.

### Assumptions

You are logged in as the oracle user.

### Tasks

##### Start an Application Workload

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Execute the $HOME/labs/PERF\_setup\_tuning.sh shell script. This script creates a user named ADMIN\_PDB1, a tablespace named TBS\_APP, and a schema named OE in the TBS\_APP tablespace. It does the same thing in PDB2, except it creates a user named ADMIN\_PDB2. You can ignore any error messages about objects not existing.

$ **$HOME/labs/PERF\_setup\_tuning.sh**

…

412129 rows created.

Commit complete. [oracle@MYDBCS ~]$

1. Start an application workload in PDB1 and PDB2.

$ **$HOME/labs/PERF\_loop.sh**

**Note:** This script generates continuous output in the terminal window where it starts.

##### Review the Performance Hub in EM Express

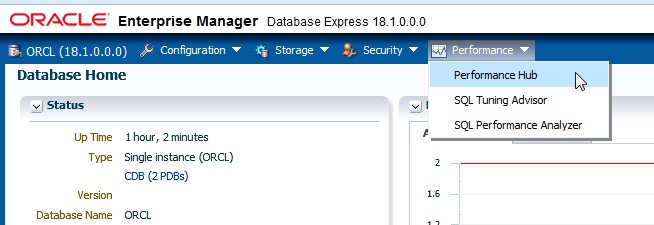
1. Open a new terminal window; create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

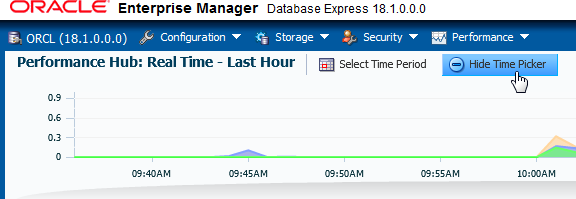
[oracle@MYDBCS ~]$

1. Open a browser. Launch Enterprise Manager Database Express by entering the following URL: https://localhost:5500/em.
2. On the Login page, enter the user name SYS and the password. Leave the Container Name box empty, select as sysdba, and click **Login**.
3. Select **Performance** and then **Performance Hub**.

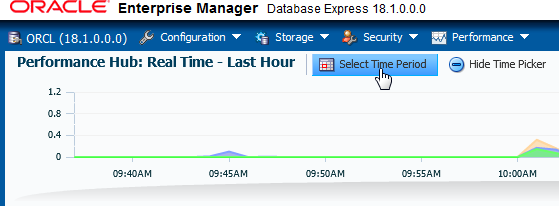


The Performance Hub provides a consolidated view of all performance data for a given time range. You must have the Oracle Diagnostics Pack (licensed option) to use the Performance Hub.

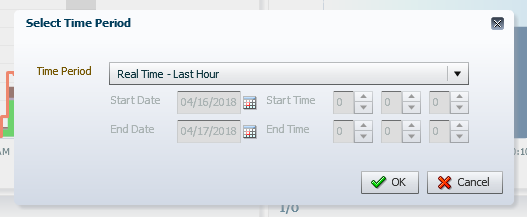
1. Learn about the Time Picker at the top of the page. The Time Picker displays average active sessions over time.
   1. Click **Hide Time Picker** and then **Show Time Picker** to hide and show it.



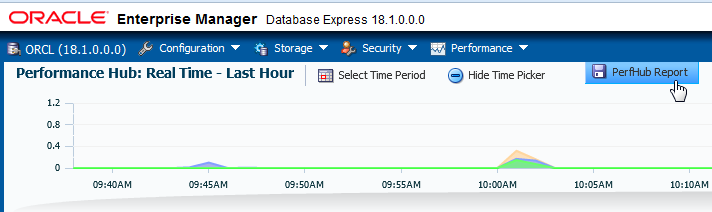
* 1. At the top of the page, click **Select Time Period**.



* 1. In the dialog box, you can select a time range, and the detail tabs will display the available performance data for the selected time range. Click the drop-down list and review the options. Notice that you can choose to view historical and real-time data.
  2. Select **Real Time - Last Hour** and click **OK**. In real-time mode, performance data is retrieved from in-memory views. The time picker shows data for the past hour and you can select any time range from within this period. The default selection is the past 5 minutes.



* 1. If there are peaks in the time picker, on the chart you can drag the selected time range to the period of interest to get more information. Drag the time picker to test this out, and try moving just one of the time picker handles to increase and decrease the time range.
  2. Click **Select Time Period**, select **Historical - Day**, and click **OK**. In historical mode, data is retrieved from the Automatic Workload Repository (AWR). You can select any time period, provided there is sufficient data in AWR. When you switch to historical mode, the default selected time range is dependent on the amount of data shown in the time picker: if the time picker displays data for the past week, the default selected time range is one day; and if the time picker displays data for the past day, the default selected time range is one hour.
  3. Change the time picker back to displaying the last hour.
  4. Click **PerfHub Report** at the top of the page.



Notice that you save the contents of the tabs with details for top SQL statements. Click Cancel.

1. The Performance Hub organizes performance data by dividing it into different tabs. Each tab addresses a specific aspect of database performance.
   1. The **Summary** tab, which is currently displayed, is available in both real-time and historical mode. In real-time mode, this tab shows metrics data that gives an overview of system performance in terms of host resource consumption (CPU, I/O, and memory) and average active sessions. In historical mode, this tab displays system performance in terms of resource consumption, average active sessions, and load profile information.
   2. Click the **Activity** tab. This tab displays Active Session History (ASH) analytics and is available in both real-time and historical mode.
   3. Click the **Workload** tab. This tab is available in both real-time and historical mode and shows metric information about the workload profile, such as call rates, logon rate, and the number of sessions. It also displays the Top SQL for the selected time range. In real-time mode, this tab displays top SQL only by database time, but in historical mode, you can also display top SQL by other metrics, such as CPU time or executions.
   4. Click the **Monitored** SQL tab. This tab displays monitored executions of SQL, PL/SQL, and database operations and is available in both real-time and historical mode.
   5. Click the **ADDM** tab. This tab displays the performance findings and recommendations of Automatic Database Diagnostic Monitor (ADDM) for database tasks performed in the selected time period. It is available in both real-time and historical mode. The ADDM analyzes data in the AWR to identify potential performance bottlenecks.
   6. Click the **Containers** tab. This tab displays performance information about each PDB in the CDB, including active sessions, memory used, I/O requests, and I/O throughput.

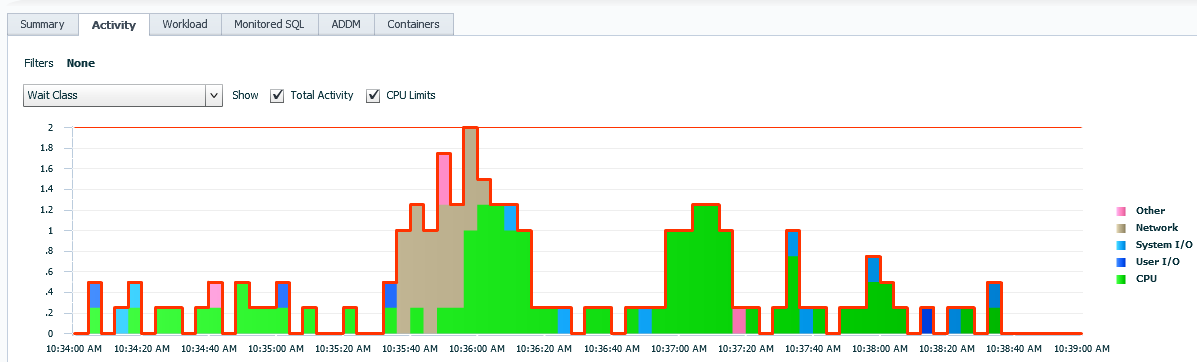
##### View Wait Statistics on the Activity Tab

On the Activity tab, you can view the Active Session History (ASH). ASH is part of the Diagnostics and Tuning Pack. It samples information from the [G]V$ views, allowing you to see current and historical information about active sessions in the database. An active session is a session that is waiting on CPU or any event that does not belong to the IDLE wait class.

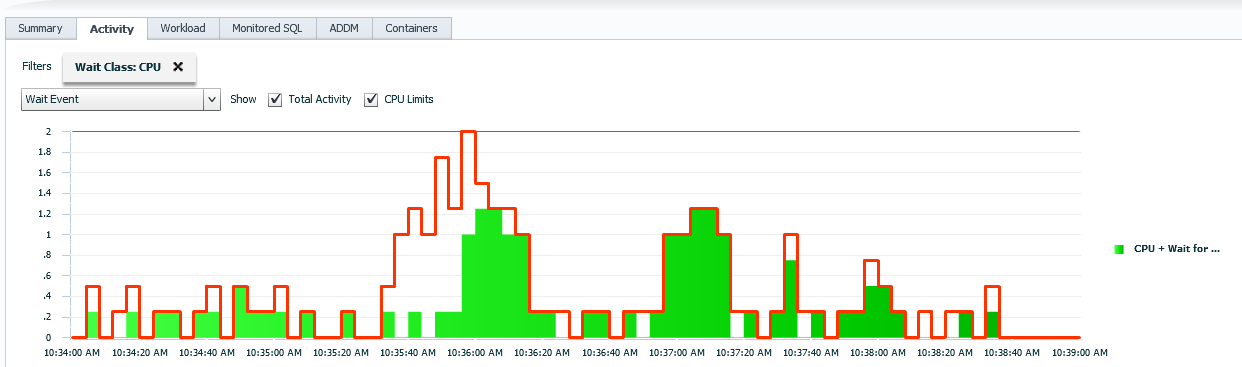
1. If your workload script has ended, start it again in the terminal window.

$ **$HOME/labs/PERF\_loop.sh**

1. In EM Express, click the Refresh button a few times. Eventually you will get some data in the time picker.
2. Click the **Activity** tab.

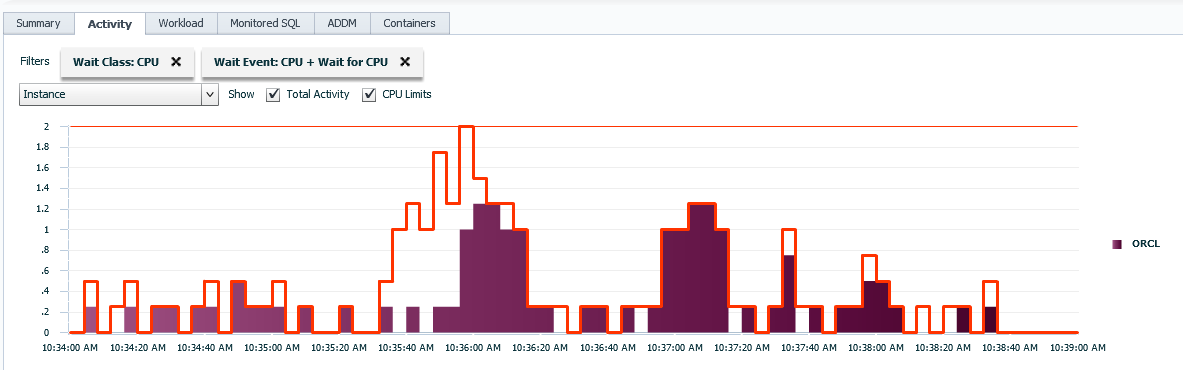


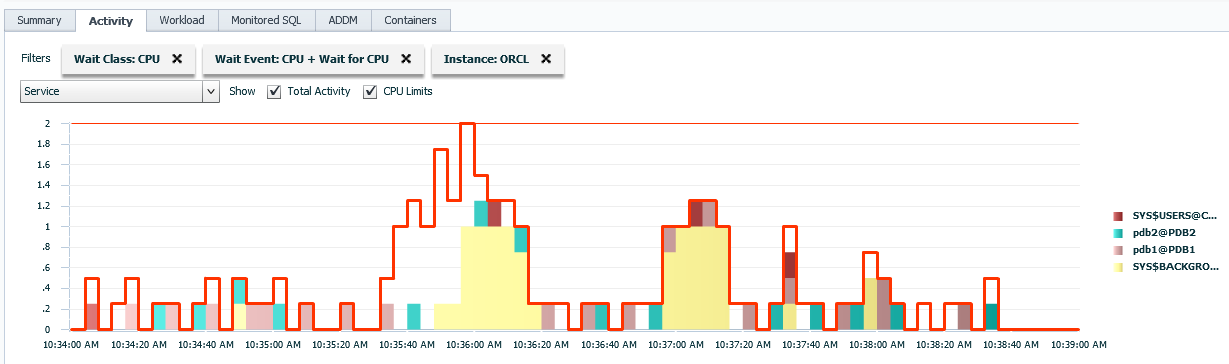
1. View the graph in the middle of the page, which shows wait event information. Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Waits and the associated SQL are key indicators for determining the root cause of an issue.
   1. By default, the graph displays the average active session waits by time, filtered by class (notice that Wait Class is selected in the filters drop-down list). Each wait class consists of wait events. For example, waits resulting from DBA commands that cause users to wait (for example, an index rebuild) are in the Administrative class. Another example is the System I/O class that consists of waits for background process IO, for example, a DBWR wait for 'db file parallel write.' The classes are listed in the legend to the right of the graph.
   2. Position your cursor over one of the wait classes in the legend, for example, the CPU class. Notice that the class is highlighted in yellow in the graph.
   3. In the legend, click the **CPU** wait class. A filter is created, and the graph drills down into the different wait events for the CPU wait class. Notice that the filters drop-down list is now Wait Event.



* 1. Click the graph. The graph is displaying only one item at the moment, so don't worry about clicking the wrong part. Notice when you positioned your cursor over the graph it

turned yellow again. Clicking the graph further drills down into wait events. So you can either click the legend items or click the graph items themselves to drill down. Now the graph displays the waits for the ORCL instance.

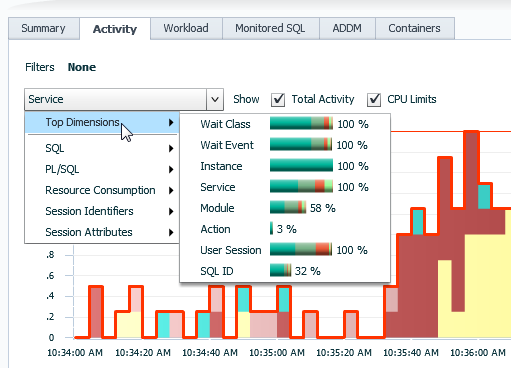


* 1. In the legend, click **ORCL**. You have now drilled down into the waits per service.
  2. Remove the filters in the graph by clicking the Xs for each filter.

uctor Use Only

.

* 1. Another way to filter the graph is to select a filter in the drop-down list. In the drop- down list, select **Top Dimensions**.



Notice that the names of the top dimensions are the same names you just saw as you drilled down into the graph through the legend, for example, Wait Class, Wait Event, Instance, Service, and so on. Selecting a top dimension is a quick and easy way to jump directly to a particular drill-down level.

* 1. Position your cursor over **Service** and view the percentage breakdown for service waits.

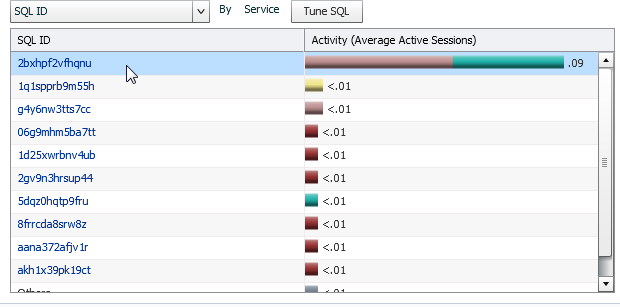
##### Filter Wait Statistics for a SQL ID

1. If your workload script has ended, start it again in the terminal window.

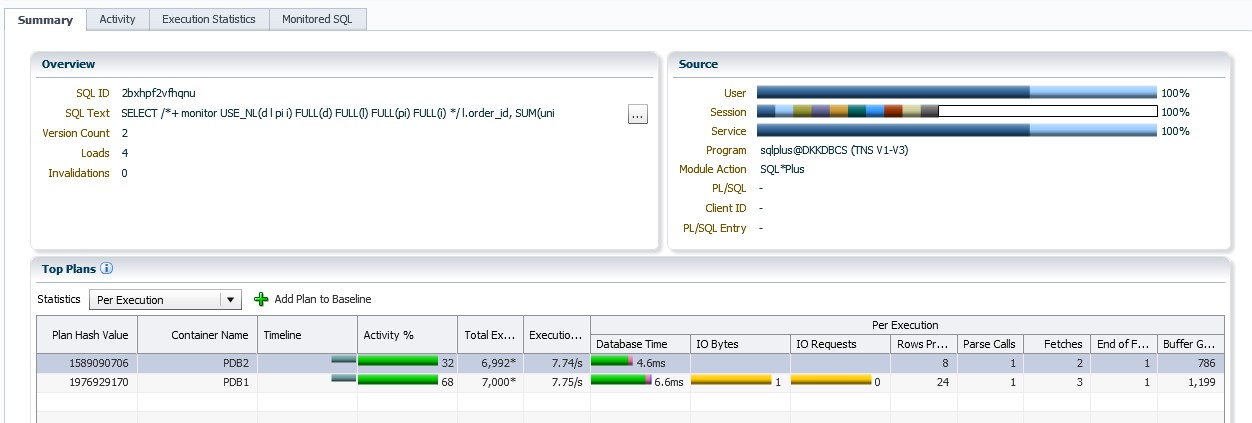
$ **$HOME/labs/PERF\_loop.sh**

1. In EM Express, at the bottom left, view the table that shows the activity (average active sessions) for each SQL ID.
   1. Click the SQL ID that has the greatest average. In this example, the SQL ID is

2bxhpf2vfhqnu.

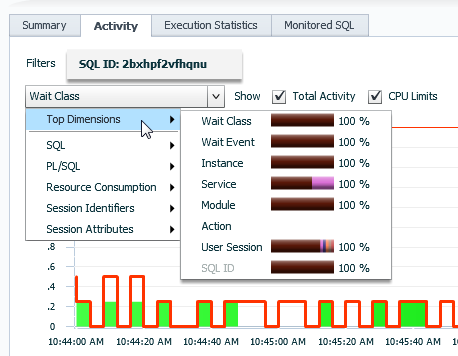


* 1. The Summary tab is displayed with performance information about the selected SQL ID.

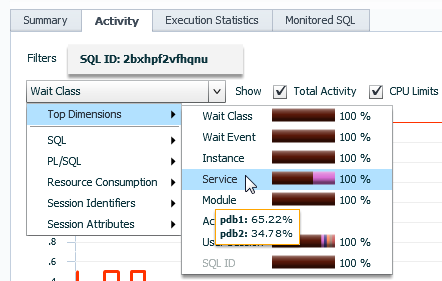
1. Click the **Activity** tab. Notice that the Activity tab is filtered based on your selected SQL ID.

For Instructor U

1. In the filters drop-down list, select **Wait Class** and then **Top Dimensions**.



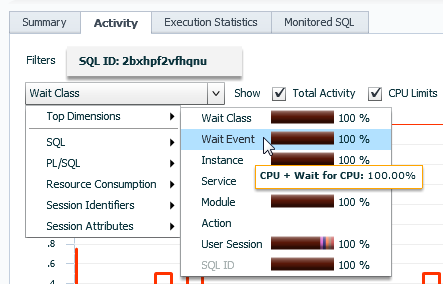
1. Position your cursor over **Service**.



1. Question: What does the information tell you?

Answer: In this example, the percentage value is higher for PDB1 (65.22%) than PDB2 (34.78%), which means that executions in PDB1 have to wait longer than in PDB2. Your values will be different.

1. Position your cursor over **Wait Event**. Notice that the execution of the statement is waiting for CPU resources.



1. Click **Log Out** to log out of EM Express, and close the browser window.
2. In the terminal window, press Crtl+C to stop the PERF\_loop.sh script.

**Practice 20-2: Resolving Lock Conflicts**

### Overview

In this practice, two users try to update data for an employee at the same time in SQL\*Plus and cause a lock conflict. Using Enterprise Manager Database Express (EM Express), you detect the blocking session and then resolve the conflict by killing the blocking session.

Note: You can also use EM Cloud Control to monitor locks conflicts.

##### Tip

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

To set a title for a terminal window:

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

### Assumptions

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

### Tasks

##### Create a Lock Conflict Window 1

1. In an open terminal window, execute the

$HOME/labs/RESOLVE\_LOCK\_ISSUES\_setup.sh shell script. You can ignore the error messages.

[oracle@MYDBCS ~]$ **$HOME/labs/RESOLVE\_LOCK\_ISSUES\_setup.sh**

drop user NGREENBERG cascade

\* ERROR at line 1:

ORA-01918: user 'NGREENBERG' does not exist

User created. Grant succeeded.

drop user SMAVRIS cascade

\* ERROR at line 1:

ORA-01918: user 'SMAVRIS' does not exist

User created. Grant succeeded. Grant succeeded.

[oracle@MYDBCS ~]$

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

[oracle@MYDBCS ~]$ **sqlplus ngreenberg/*password*@PDB1**

…

SQL>

1. Update the HR.EMPLOYEES table, but do not commit or exit the SQL\*Plus session. The following statement updates the phone number to 650.555.1212 for employee ID 110. This session will be referred to as the "blocking session."

SQL> **UPDATE hr.employees SET phone\_number='650.555.1212' WHERE employee\_id = 110;**

1 row updated.

SQL>

##### Window 2

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

[oracle@MYDBCS ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Start SQL\*Plus and connect as SMAVRIS. Refer to *Course Practice Environment: Security Credentials* for the password value.

[oracle@MYDBCS ~]$ **sqlplus smavris/*password*@PDB1**

…

SQL>

1. Update the HR.EMPLOYEES table. The following statement updates the salary to 8300 for employee ID 110.

SQL> **UPDATE hr.employees SET salary = 8300 WHERE employee\_id = 110;**

Notice that the session in Window 2 is hung. This session will be referred to as the "blocked session." Leave this session as is and move on to the next step.

1. Question: Which situation can cause lock conflicts?

Answer: The most common cause of lock conflicts is an uncommitted change, but there are a few other possible causes, such as:

* + Long-running transactions: Many applications use batch processing to perform bulk updates. These batch jobs are usually scheduled for times of low or no user activity, but in some cases, they may not have finished or may take too long to run during the low activity period. Lock conflicts are common when transaction and batch processing are being performed simultaneously.
  + Unnecessarily high locking levels: Not all databases support row-level locking (Oracle added support for row-level locks in 1988 with release 6). Some databases still lock at the page or table level. Developers writing applications that are intended to run on many different databases often write their applications with artificially high locking levels so that Oracle Database behaves similarly to these less capable database systems. Developers who are new to Oracle also sometimes unnecessarily code in higher locking levels than are required by Oracle Database.

1. SMAVRIS, who is connected in Window 2, informs you that his transaction is blocked. Question: What is the best way to fix the locking conflict?

Answer: The solution is to have the session release the lock. Contact the user and ask that the transaction be completed. In an emergency, it is possible for the administrator to terminate the session holding the lock. Remember that when a session is killed, all work within the current transaction is lost (rolled back). A user whose session is killed must log in again and redo all work since the killed session’s last commit.

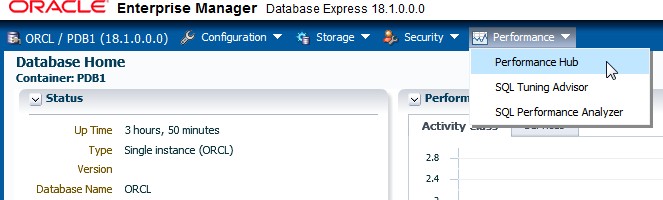
##### Use Enterprise Manager Database Express to Release the Lock

In this section, you use EM Express to locate the blocking session and retrieve the identifiers that enable you to kill the blocking session.

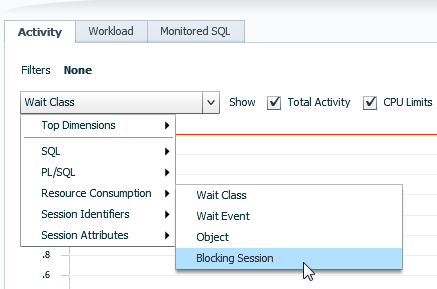
1. Open a browser. Launch Enterprise Manager Database Express by entering the following URL: https://localhost:5500/em
2. On the Login page for EM Express, enter the user name **SYS** and the password. Enter

**PDB1** as the container name. Select **as sysdba**. Click **Login**.

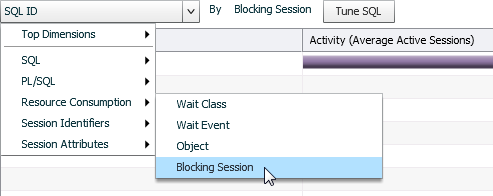
1. Select **Performance** and then **Performance Hub**.



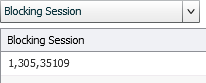
1. The Activity tab is displayed by default. In the filter drop-down list, select Resource Consumption and then Blocking Session.



1. In the bottom drop-down list, also select Resource Consumption and then Blocking Session to retrieve the identifiers that allow you to kill the blocking session.



1. In the Blocking Session column at the bottom of the page, make note of the second number (SID) and third number (serial number). The blocking session is identifiable by its SID and serial number, and you will use this information later in SQL\*Plus. In this example, the SID is 305 and the serial number is 35109. Your numbers will be different.



Be aware that if, in the bottom drop-down list, you select User Session (select Top Dimensions and then User Session), you will get the list of *blocked sessions* and not the session to kill.

1. Log out of EM Express.

##### Terminate the Blocking Session

Now that you know the SID and serial number, you can terminate the blocking session.

##### Window 3

1. Open a new terminal window and connect to the compute node as the oracle user. This window will be referred to as Window 3.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

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

[oracle@MYDBCS ~]$ **sqlplus SYSTEM/*password*@PDB1**

…

SQL>

1. Kill the session using the SID and serial number that you identified in EM Express. Your numbers will be different from those shown below.

SQL> **ALTER SYSTEM KILL SESSION '305,35109';**

System altered.

SQL>

1. Log out of SQL\*Plus and close the Window 3 terminal window.

SQL> **exit**

…

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

##### Window 2

1. In Window 2, find out what happened to the blocked session. Verify that the salary was updated.
   1. Notice that in window 2, the blocked session is now released and the result is "1 row updated."

1 row updated.

SQL>

* 1. Query the HR.EMPLOYEES table to find out whether the salary was updated for employee ID 110. The result shows that the salary was updated to 8300.

SQL> **SELECT phone\_number , salary FROM hr.employees WHERE employee\_id = 110;**

-------------------- ---------- 515.124.4269 8300

SQL>

SALARY

PHONE\_NUMBER

* 1. Roll back the salary update because you don't need to keep the update.

SQL> **ROLLBACK;**

Rollback complete. SQL>

1. Log out of SQL\*Plus and close the Window 2 terminal window.

SQL> **exit**

…

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

##### Window 1

1. In Window 1, find out what happened to the blocking session. Query the HR.EMPLOYEES

table to find out the phone number for employee ID 110.

SQL> **SELECT phone\_number FROM hr.employees WHERE employee\_id = 110;**

SELECT phone\_number FROM hr.employees WHERE employee\_id = 110

\* ERROR at line 1:

ORA-00028: your session has been killed

SQL>

The result indicates that the session was killed. The locks were released, and the update was rolled back.

1. Log out of SQL\*Plus and close the Window 1 terminal window.

SQL> **exit**

…

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

**Practices for Lesson 21: SQL**

**Tuning**

**Practices for Lesson 21: Overview**

### Overview

In these practices, you will use the SQL Tuning Advisor to optimize SQL performance.

**Practice 21-1: Using the SQL Tuning Advisor**

### Overview

In this practice, you optimize the performance of a costly SQL statement by using the SQL Tuning Advisor through Enterprise Manager Database Express (EM Express).

Note: All advisors are also available through Enterprise Manager Cloud Control.

### Assumptions

You are logged in as the oracle user.

### Tasks

##### Initiate a SQL Load

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

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

1. Source the oraenv script.

[oracle@MYDBCS ~]$ **. oraenv**

ORACLE\_SID = [ORCL] ?

The Oracle base remains unchanged with value /u01/app/oracle

[oracle@MYDBCS ~]$

1. Execute the $HOME/labs/PERF\_setup\_tuning.sh shell script. This script creates a user named ADMIN\_PDB1, a tablespace named TBS\_APP, and a schema named OE in the TBS\_APP tablespace in PDB1. It does the same thing in PDB2, except it creates a user named ADMIN\_PDB2. Wait for the setup script to finish. It may take a couple of minutes to run. You can ignore any error messages because they are expected.

[oracle@MYDBCS ~]$ **$HOME/labs/PERF\_setup\_tuning.sh**

…

412129 rows created.

Commit complete. [oracle@MYDBCS ~]$

1. Start an application workload in PDB1 and PDB2. The PERF\_loop.sh script runs a SQL script named PERF\_loop.sql eight times in PDB1 as the OE user. It then runs the same SQL script eight times in PDB2 as the SYSTEM user. You can move on to the next step while the script is running.

[oracle@MYDBCS ~]$ **$HOME/labs/PERF\_loop.sh**

…

##### Use EM Express to Tune the SQL Based on Statistics

In this section, you review but do not implement the first recommendation of the SQL Tuning Advisor, which is based on statistics.

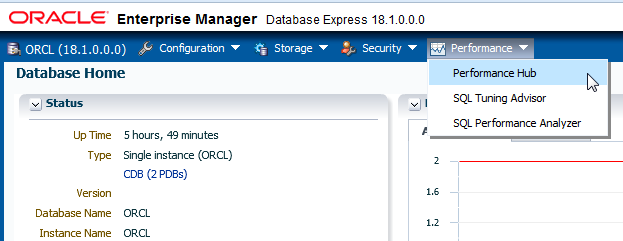
1. Open a new terminal window and create an SSH tunnel.

[oracle@edvm ~]$ **cd ~/.ssh**

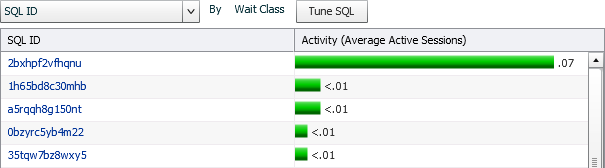
[oracle@edvm ~]$ **ssh -i *your\_private\_key\_file* -L 5500:*your\_compute\_node\_IP\_address*:5500 oracle@*your\_compute\_node\_IP\_address***

[oracle@MYDBCS ~]$

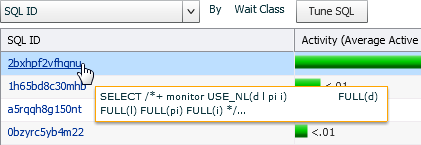
1. Open a browser and launch Enterprise Manager Database Express by entering the following URL: https://localhost:5500/em
2. On the Login page, enter the user name **SYS** and the password. Leave the Container Name box empty, select **as sysdba**, and click **Login**.
3. Select **Performance** and then **Performance Hub**.



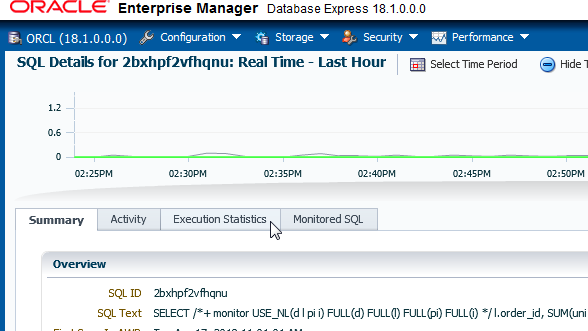
1. Click the **Activity** tab. The current SQL executions are listed in the table at the bottom of the page. In this example, you can see that there is one consuming SQL execution (the first SQL ID in the list).



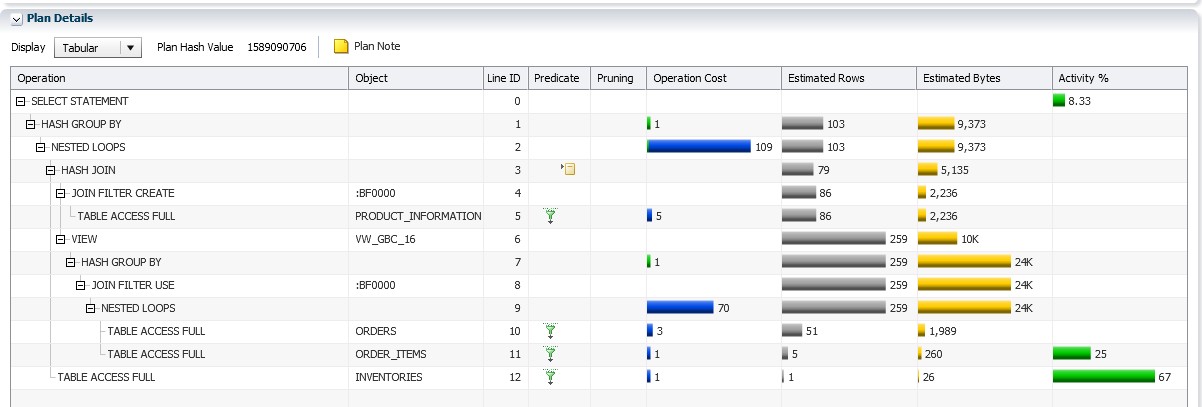
1. Position your cursor over the SQL ID. The following code should appear. If your result looks different, wait a moment and refresh EM Express.



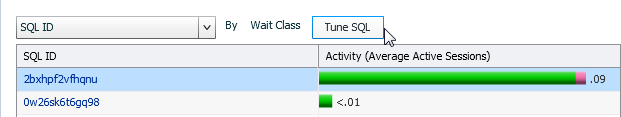
1. Click the SQL ID, and then click the **Execution Statistics** tab.



1. The Plan Details area at the bottom of the page shows you the current plan for executing the SQL.



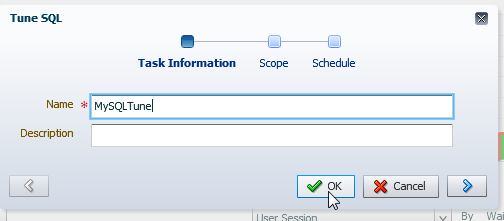
1. To return to where you were last, select **Performance** and then **Performance Hub**. Click the **Activity** tab.
2. In the Activity column, click the row to select the SQL statement to tune, and click the **Tune SQL** button to launch the SQL Tuning Advisor.



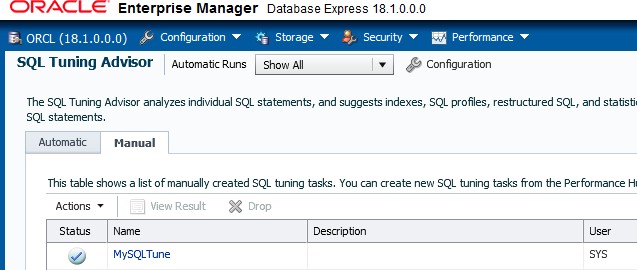
1. Question: What may the SQL Tuning Advisor suggest?

Answer: It can suggest indexing columns, SQL profiles implementation, restructuring the SQL statement, and collecting missing or stale object statistics.

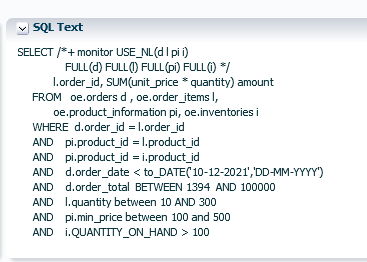
1. In the Tune SQL dialog box, enter the task name **MySQLTune** and click **OK**.



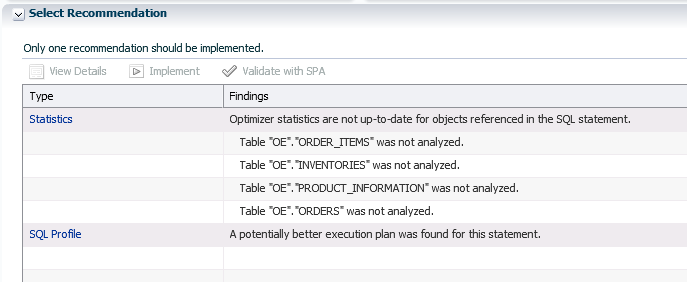
1. While the analysis task is completing, you are automatically brought to the SQL Tuning Advisor page, which has two tabs:
   * The Automatic tab lists the automatic tasks executed every night.
   * The Manual tab lists the manually created SQL tuning tasks.
2. Click the **Manual** tab. The task you just created is listed. You may need to wait a moment for it to complete its processing.



1. Click your task to read the recommendations.
2. In the SQL Text area, you can view the full SQL statement to be tuned. Scroll down to view all of it.



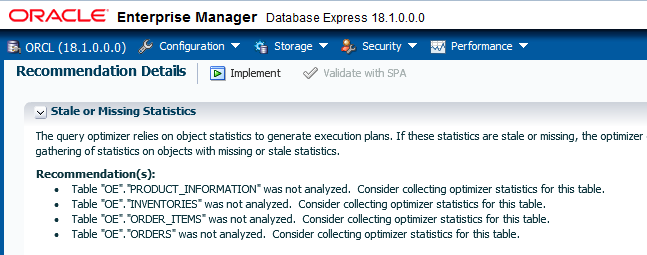
1. In the **Select Recommendation** area, notice that there are two recommendations: Statistics and SQL Profile. For the Statistics recommendation, SQL Tuning Advisor's findings say that Optimizer statistics are not up-to-date for objects referenced in the SQL statement. It identifies OE tables in the SQL statement that are not analyzed.



1. Question: How do tables get statistics collected automatically?

Answer: There is an automated task that automatically gathers Optimizer statistics every night. You can configure the settings that are used for Optimizer statistics gathering.

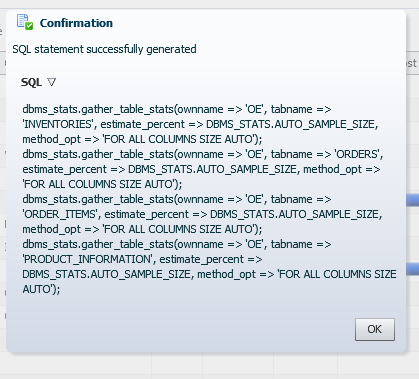
1. In the **Select Recommendations** area, click the **Statistics** link.
2. On the Recommendation Details page, view the list of recommendations, and then click the **Implement** button at the top of the page. Notice that the Optimizer recommends gathering statistics for the tables in your SQL statement.



1. In the Gather Statistics dialog box, click **Show SQL**.



1. View the SQL package and procedure that would gather the statistics, and click **OK**.

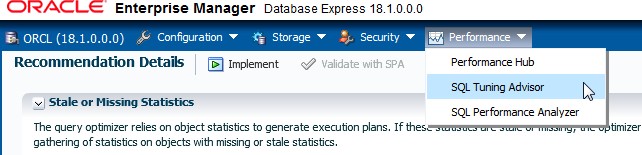


1. In the Gather Statistics dialog box, click Cancel because you will ask the Optimizer Advisor in the next practice to help you implement the statistics collection in the best way.

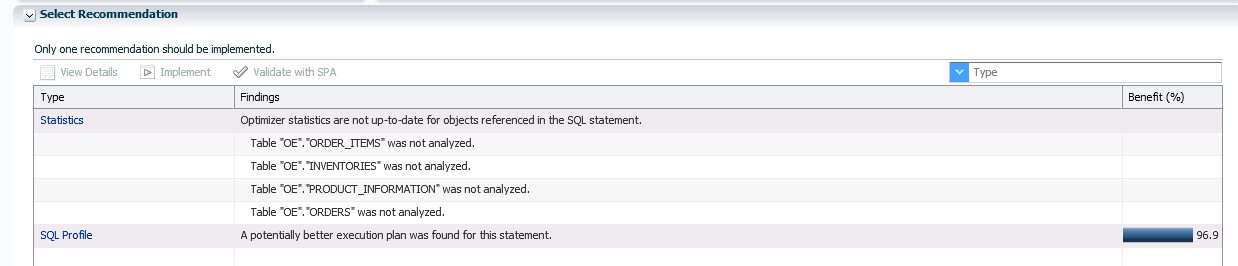
##### Use EM Express to Tune the SQL Using a SQL Profile

In this section, you implement the second recommendation, which suggests the usage of a SQL profile. This option provides a better execution plan.

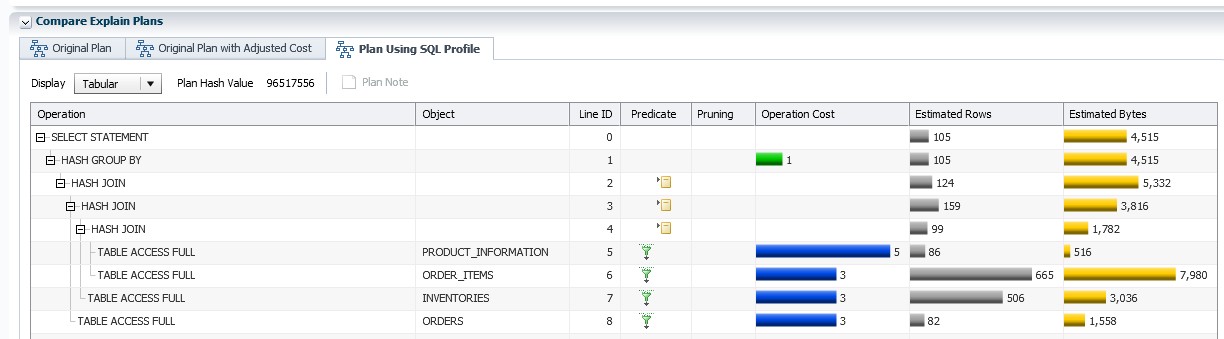
1. Select **Performance** and then **SQL Tuning Advisor**.



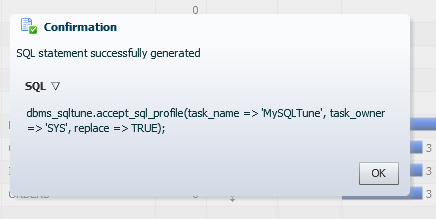
1. Click the **Manual** tab, if needed.
2. Click **MySQLTune** (or the name you gave your tuning task).
3. In the **Select Recommendation** section, at the bottom of the Benefit column, view the benefit percentage value for using the SQL Profile. In this example, the SQL profile would increase performance by almost 97%. Your value may be different.



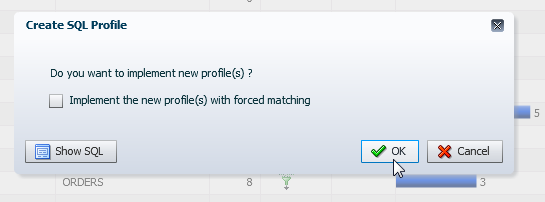
1. In the Select Recommendation section, click the SQL Profile link.
2. On the Original Plan tab, which is displayed by default, view the SQL execution plan. You saw this plan earlier in the practice.
3. Click the Plan Using SQL Profile tab and view its SQL execution plan. Notice the differences between it and the original plan.



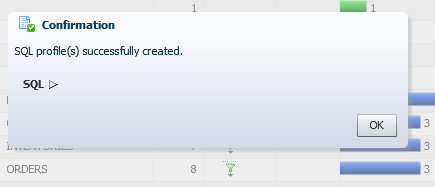
1. At the top of the page, click the **Implement** button.
2. In the Create SQL Profile dialog box, click **Show SQL**.
3. A Confirmation dialog box shows you the generated SQL statement. Click **OK**.



1. In the Create SQL Profile dialog box, click **OK** to implement the new profile.



1. The SQL profile is created. In the Confirmation dialog box, click **OK**.



##### Rerun the SQL Script and Verify the Performance Benefit

In this section, you re-execute the PERF\_loop.sh script and verify that the SQL tuning you just implemented made the query consume fewer resources in the database.

1. Return to the terminal window.
2. If the script is still running, press Ctrl+c to stop the activity.
3. Start SQL\*Plus and connect to the CDB root as the SYS user with the SYSDBA privilege.

[oracle@MYDBCS ~]$ **sqlplus / AS SYSDBA**

…

SQL>

1. When testing SQL, it is a good idea to periodically flush the shared pool entries to remove older execution plans.

SQL> **ALTER SYSTEM FLUSH SHARED\_POOL;**

System altered.

SQL>

1. Remove any blocks of the tables (selected in the query) from the buffer cache.

SQL> **ALTER SYSTEM FLUSH BUFFER\_CACHE;**

System altered.

SQL>

1. Exit SQL\*Plus.

SQL> **EXIT**

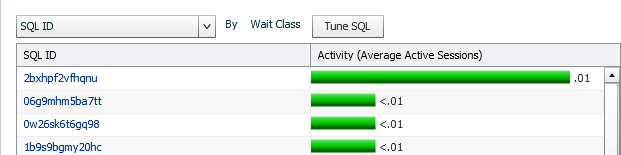
…

[oracle@MYDBCS ~]$

1. Run the application workload again in PDB1 and PDB2. The PERF\_loop.sh script runs a SQL script named PERF\_loop.sql eight times in PDB1 as the OE user. It then runs the same SQL script eight times in PDB2 as the SYSTEM user.

[oracle@MYDBCS ~]$ **$HOME/labs/PERF\_loop.sh**

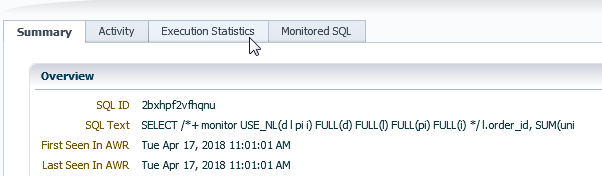
1. Return to EM Express.
2. If a Warning dialog box is displayed stating that a particular SQL ID is no longer from Cursor Cache, click **OK**.
3. Select **Performance** and then **Performance** Hub.
4. Click the **Activity** tab.
5. At the bottom of the page, note the value in the Activity column.



1. Question: How does the value in the Activity column now compare to the value in the Activity column prior to SQL tuning? Is there a performance benefit to the SQL tuning that you just did?

Answer: In this example, the average active sessions value went down from .07 to .01, so yes, there is a performance benefit from tuning the SQL. Your values may differ.

1. Click the link for the SQL ID. The Summary tab is displayed for the SQL ID.
2. Click the **Execution Statistics** tab.



1. In the Plan Details area, notice that the plan now used is the plan that uses the SQL Profile (refer back to step 7 in the previous section in this practice).
2. Click **Log Out** to exit EM Express and close the browser window.
3. In the terminal window, press Ctrl+c to stop the activity.

**Practice 21-2: Using the Optimizer Statistics Advisor**

### Overview

In this practice, you learn how to improve optimizer statistics collection quality by using the Optimizer Statistics Advisor.

The advisor task runs automatically in the maintenance window, but you can also run it on demand. If the advisor makes findings and then recommendations, then in some cases you can run system-generated scripts to implement them. Optimizer statistics play a significant part in determining the execution plan for queries. Therefore, it is critical for the optimizer to gather and maintain accurate and up-to-date statistics. All findings are derived from rules, but not all rules generate findings.

##### Tip

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

To set a title for a terminal window:

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

### Assumptions

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

### Tasks

##### Window 1: Start an Application Workload

1. Execute the $HOME/labs/PERF\_setup\_tuning.sh shell script. Wait for the setup script to finish. You can ignore any error messages because they are expected.

[oracle@MYDBCS ~]$ **$HOME/labs/PERF\_setup\_tuning.sh**

…

412129 rows created.

Commit complete. [oracle@MYDBCS ~]$

1. Start an application workload in PDB1 and PDB2.

$ **$HOME/labs/PERF\_loop.sh**

##### Window 2: Use the Optimizer Statistics Advisor to Generate Recommendations

In this section, you create an object filter for an Optimizer Statistics Advisor task, create and execute the task, generate a report with recommendations, and then implement those recommendations.

1. Open another new terminal window and connect to the compute node. This window will be referred to as Window 2.

[oracle@edvm ~]$ **cd ~/.ssh**

[oracle@edvm .ssh]$ **ssh -i *your\_private\_key\_file***

**oracle@*your\_compute\_node\_IP\_Address***

[oracle@MYDBCS ~]$

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

[oracle@MYDBCS ~]$ **sqlplus SYS/*password*@PDB1 AS SYSDBA**

..

SQL>

1. Execute the $HOME/labs/OPTADV\_1.sql script, which is an object filter for an Optimizer Advisor Task. This filter disables statistics collection recommendations for all objects except those in the OE schema. In Practice 21-1 Using SQL Tuning Advisor, the OE tables were included in queries that required tuning.

CREATE OR REPLACE PROCEDURE sh\_obj\_filter(p\_tname IN VARCHAR2) IS v\_retc CLOB;

BEGIN

v\_retc := DBMS\_STATS.CONFIGURE\_ADVISOR\_OBJ\_FILTER (p\_tname,'EXECUTE',NULL,NULL,NULL,'DISABLE'); v\_retc := DBMS\_STATS.CONFIGURE\_ADVISOR\_OBJ\_FILTER (p\_tname,'EXECUTE',NULL,'OE',NULL,'ENABLE');

END;

/

SQL> **@$HOME/labs/OPTADV\_1.sql**

Procedure created. SQL>

1. Create and execute an advisor task named my\_task by executing the

$HOME/labs/OPTADV\_2.sql script.

DECLARE

v\_tname VARCHAR2(128) := 'my\_task'; v\_ename VARCHAR2(128) := NULL;

v\_report CLOB := null; v\_script CLOB := null;

v\_implementation\_result CLOB; BEGIN

v\_tname := DBMS\_STATS.CREATE\_ADVISOR\_TASK(v\_tname); sh\_obj\_filter(v\_tname);

v\_ename := DBMS\_STATS.EXECUTE\_ADVISOR\_TASK(v\_tname); END;

/

SQL> **@$HOME/labs/OPTADV\_2.sql**

PL/SQL procedure successfully completed. SQL>

1. Verify that the procedure completed successfully.
   1. Query the USER\_ADVISOR\_TASKS view.

-------------------- --------------- --------------- -----------

Statistics Advisor STATISTICS EXEC\_31 COMPLETED

SQL>

EXECUTION\_TYPE LAST\_EXECUTION STATUS

ADVISOR\_NAME

SQL> **col advisor\_name format a20** SQL> **col execution\_type format a15** SQL> **col last\_execution format a15** SQL> **col status format a15**

SQL> **SELECT advisor\_name, execution\_type, last\_execution, status**

1. **FROM user\_advisor\_tasks**
2. **WHERE task\_name = 'MY\_TASK';**
   1. Query the USER\_ADVISOR\_EXECUTIONS view. The results below are formatted for easier viewing. Your dates will be different from those shown below. Make note of the value in the EXECUTION\_NAME column for MY\_TASK. In this example, the value is EXEC\_31.

07-FEB-18 STATISTICS

EXEC\_1

AUTO\_STATS\_ADVISOR\_TASK COMPLETED

------------------------- ---------- --------- --------------- -

-------------

EXECUTION\_ EXECUTION TYPE

TASK\_NAME STATUS

SQL> **SELECT task\_name, execution\_name, execution\_end, execution\_type AS type, status**

2 **FROM user\_advisor\_executions;**

|  |  |  |  |
| --- | --- | --- | --- |
| MY\_TASK COMPLETED | EXEC\_31 | 18-APR-18 | STATISTICS |
| SYS\_AUTO\_SPM\_EVOLVE\_TASK COMPLETED | EXEC\_11 | 10-APR-18 | SPM EVOLVE |
| AUTO\_STATS\_ADVISOR\_TASK COMPLETED | EXEC\_21 | 10-APR-18 | STATISTICS |
| SQL> |  |  |  |
| 6. Generate a report. |  |  |  |

1. Execute the $HOME/labs/OPTADV\_3.sql script to run the following commands:

VAR b\_report CLOB DECLARE

v\_tname VARCHAR2(32767);

BEGIN

v\_tname := 'my\_task';

:b\_report := dbms\_stats.report\_advisor\_task(v\_tname, type => 'TEXT', section=>'ALL', level=>'ALL');

END;

/

SQL> **@$HOME/labs/OPTADV\_3.sql**

PL/SQL procedure successfully completed. SQL>

1. Execute the $HOME/labs/OPTADV\_4.sql script to run the following commands:

DECLARE

v\_len NUMBER(10); v\_offset NUMBER(10) :=1;

v\_amount NUMBER(10) :=10000;

BEGIN

v\_len := DBMS\_LOB.getlength(:b\_report); WHILE (v\_offset < v\_len)

LOOP

DBMS\_OUTPUT.PUT\_LINE(DBMS\_LOB.SUBSTR(:b\_report,v\_amount,v\_offset

));

v\_offset := v\_offset + v\_amount; END LOOP;

END;

/

SQL> **@$HOME/labs/OPTADV\_4.sql**

PL/SQL procedure successfully completed.

SQL>

1. Execute the $HOME/labs/OPTADV\_5.sql script to view the findings. Be sure to enter the correct value for F.EXECUTION\_NAME as determined in step 5b.

SELECT f.finding\_id, f.message, r.benefit\_type

FROM user\_advisor\_findings f, user\_advisor\_recommendations r

WHERE f.finding\_id = r.finding\_id AND f.task\_name = 'MY\_TASK' AND f.execution\_name = 'EXEC\_31';

SQL> **@$HOME/labs/OPTADV\_5.sql**

FINDING\_ID

---------- MESSAGE

---------------------------------------------------------------- BENEFIT\_TYPE

---------------------------------------------------------------- 1

There are 2 statistics operation(s) using nondefault parameters. Set parameter job\_queue\_processes to 1 or higher.

2

There are 1 uses of GATHER\_TABLE\_STATS.

Set parameter \_enable\_automatic\_maintenance to 1.

3

There are 10 object(s) with no statistics. Turn on SQL Plan Directives.

1

There are 2 statistics operation(s) using nondefault parameters. Use default parameters for statistics operations.

2

There are 1 uses of GATHER\_TABLE\_STATS.

Use GATHER\_SCHEMA\_STATS instead of GATHER\_TABLE\_STATS.

3

There are 10 object(s) with no statistics.

Gather Statistics on those objects with no statistics.

6 rows selected.

SQL>

1. Generate the script before a possible implementation.
   1. Execute the $HOME/labs/OPTADV\_6.sql script to run the following commands:

SET SERVEROUTPUT ON

VARIABLE b\_script CLOB DECLARE

v\_tname VARCHAR2(32767);

BEGIN

v\_tname := 'my\_task';

:b\_script := DBMS\_STATS.SCRIPT\_ADVISOR\_TASK(v\_tname); END;

/

SQL> **@$HOME/labs/OPTADV\_6.sql**

PL/SQL procedure successfully completed. SQL>

* 1. Execute the $HOME/labs/OPTADV\_7.sql script to run the following commands:

DECLARE

v\_len NUMBER(10); v\_offset NUMBER(10) :=1;

v\_amount NUMBER(10) :=10000;

BEGIN

v\_len := DBMS\_LOB.getlength(:b\_report); WHILE (v\_offset < v\_len)

LOOP

DBMS\_OUTPUT.PUT\_LINE(DBMS\_LOB.SUBSTR(:b\_script, v\_amount, v\_offset));

v\_offset := v\_offset + v\_amount; END LOOP;

END;

/

SQL> **@$HOME/labs/OPTADV\_7.sql**

-- Script generated for the recommendations from execution EXEC\_31

-- in the statistics advisor task MY\_TASK

-- Script version 12.2

…

-- Scripts for rule AVOIDSTALESTATS

-- Rule Description: Avoid objects with stale or no statistics

-- Gather statistics for those objcts that are missing or have no statistics.

-- Scripts for rule MAINTAINSTATSCONSISTENCY

-- Rule Description: Statistics of dependent objects should be consistent

-- Gather statistics for those objcts that are missing or have no statistics.

declare

obj\_filter\_list dbms\_stats.ObjectTab; obj\_filter dbms\_stats.ObjectElem; obj\_cnt number := 0;

begin

obj\_filter\_list := dbms\_stats.ObjectTab(); obj\_filter.ownname := 'OE';

obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'CUSTOMER'; obj\_filter\_list.extend();

obj\_cnt := obj\_cnt + 1; obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'CUSTOMERS';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter;

obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'DATE\_DIM';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'INVENTORIES'; obj\_filter\_list.extend();

obj\_cnt := obj\_cnt + 1; obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE';

obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'LINEORDER';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'ORDERS'; obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE';

obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'ORDER\_ITEMS';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'PART';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE';

obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'PRODUCT\_INFORMATION'; obj\_filter\_list.extend();

obj\_cnt := obj\_cnt + 1; obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'SUPPLIER'; obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter;

dbms\_stats.gather\_database\_stats( obj\_filter\_list=>obj\_filter\_list);

end;

/

PL/SQL procedure successfully completed.

SQL>

1. Question: What does obj\_filter\_list indicate?

Answer: The object filter list contains the names of the ten objects with no statistics.

1. Question: What would you do if you agree with the recommendations? Answer: You could either execute the generated SQL script or use the

DBMS\_STATS.IMPLEMENT\_ADVISOR\_TASK procedure.

1. Execute the $HOME/labs/OPTADV\_8.sql script to invoke the DBMS\_STATS.IMPLEMENT\_ADVISOR\_TASK PL/SQL procedure. This procedure implements the actions recommended by the advisor based on results from a specified Optimizer Statistics Advisor execution.

VARIABLE b\_ret CLOB DECLARE

v\_tname VARCHAR2(32767);

BEGIN

v\_tname := 'my\_task';

:b\_ret := DBMS\_STATS.IMPLEMENT\_ADVISOR\_TASK(v\_tname); END;

/

SQL> **@$HOME/labs/OPTADV\_8.sql**

PL/SQL procedure successfully completed. SQL>

1. Check that the statistics are collected for the ten objects. Columns such as NUM\_ROWS, EMPTY\_BLOCKS, BLOCKS, and AVG\_ROW\_LEN have null values until the statistics are collected.

0

5

665

0

5

105

ORDERS

38 18-APR-18

ORDER\_ITEMS 18 18-APR-18

-------------------- ---------- ---------- ------------

----------- ---------

BLOCKS EMPTY\_BLOCKS

NUM\_ROWS

TABLE\_NAME AVG\_ROW\_LEN LAST\_ANAL

SQL> **COL table\_name FORMAT A20**

SQL> **SELECT table\_name, num\_rows, blocks, empty\_blocks, avg\_row\_len, last\_analyzed**

2 **FROM dba\_tables WHERE owner='OE';**

|  |  |  |  |
| --- | --- | --- | --- |
| CUSTOMERS | 319 | 13 | 0 |
| 158 18-APR-18 |  |  |  |
| INVENTORIES 11 18-APR-18 | 636 | 5 | 0 |
| PRODUCT\_INFORMATION 216 18-APR-18 | 287 | 13 | 0 |
| LINEORDER  98 18-APR-18 | 3297032 | 47229 | 0 |
| PART  87 18-APR-18 | 1600000 | 20645 | 0 |
| SUPPLIER  102 18-APR-18 | 16000 | 260 | 0 |
| CUSTOMER  107 18-APR-18 | 240000 | 3845 | 0 |
| DATE\_DIM  100 18-APR-18 | 2556 | 43 | 0 |
| 10 rows selected. |  |  |  |
| SQL> |  |  |  |

1. Drop the advisor task.

SQL> **exec DBMS\_STATS.DROP\_ADVISOR\_TASK('my\_task')**

PL/SQL procedure successfully completed. SQL>

1. Exit SQL\*Plus and close Window 2.

SQL> **exit**

…

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

##### Window 1: Stop the Workload

1. In Window 1, press Ctrl+c to stop the activity and then close the window.