

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.....
Practice 8-1: Exploring the Default Listener
Practice 8-2: Creating a Static Listener for a PDB
Practice 8-3: Verifying the Net Service Name for PDB1.....
Practices for Lesson 9: Administering User Security.....
Practices for Lesson 9: Overview.....
Practice 9-1: Creating Common and Local Users.....
Practice 9-2: Creating a Local User for an Application
Practice 9-3: Granting a Local Role (DBA) 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.....
Practice 9-7: Configuring a Default Role for a User
Practice 9-8: Exploring OS and Password File Authentication
Lab 10: Creating PDBs.....
Practices for Lesson 10: Overview.....
Practice 10-1: Creating a PDB from Seed.....
Practice 10-2: Cloning a PDB
Practice 10-3: Unplugging and Plugging in a PDB
Practice 10-4: Dropping a PDB.....
Lab 11: Creating Master Encryption Keys for PDBs.....
Practices for Lesson 11: Overview.....
Practice 11-1: Creating and Activating an Encryption Key
Practice 11-2: Creating and Activating the Encryption Key for PDB2.....
Lab 12: Creating and Managing Tablespaces
Practices for Lesson 12: Overview.....
Practice 12-1: Viewing Tablespace Information
Practice 12-2: Creating a Tablespace
Practice 12-3: Creating a Tablespace that is Encrypted by Default
Lab 13: Managing Storage Space
Practices for Lesson 13: Overview.....
Practice 13-1: Managing Space in Tablespaces.....
Practice 13-2: Using Compression
Practice 13-3: Enabling the Resumable Space Allocation Feature
Lab 14: Managing Undo Data.....
Practices for Lesson 14: Overview.....
Practice 14-1: Managing Undo Data
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 WholeDatabase 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:

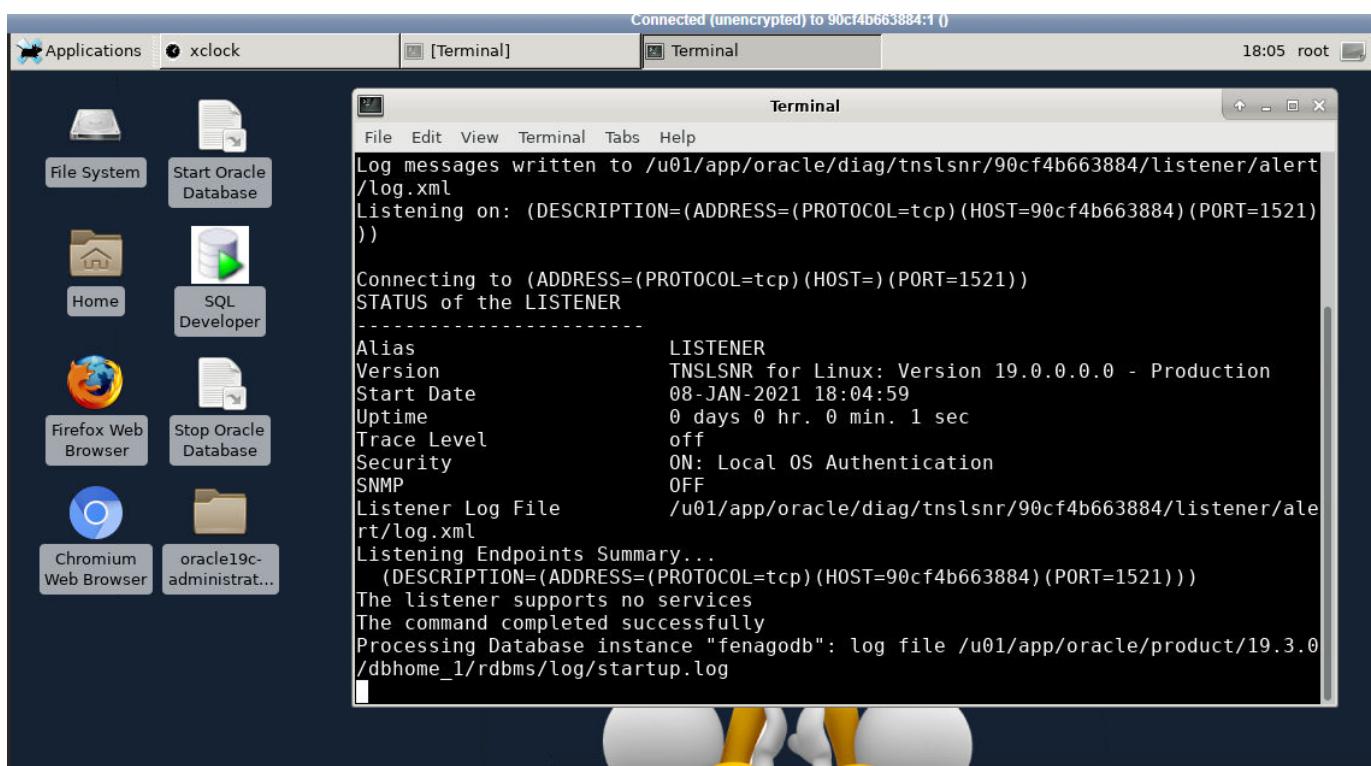
ORACLE_SID=fenagodb

ORACLE_PDB=fenagodb1

ORACLE_PWD=fenago

Tasks

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



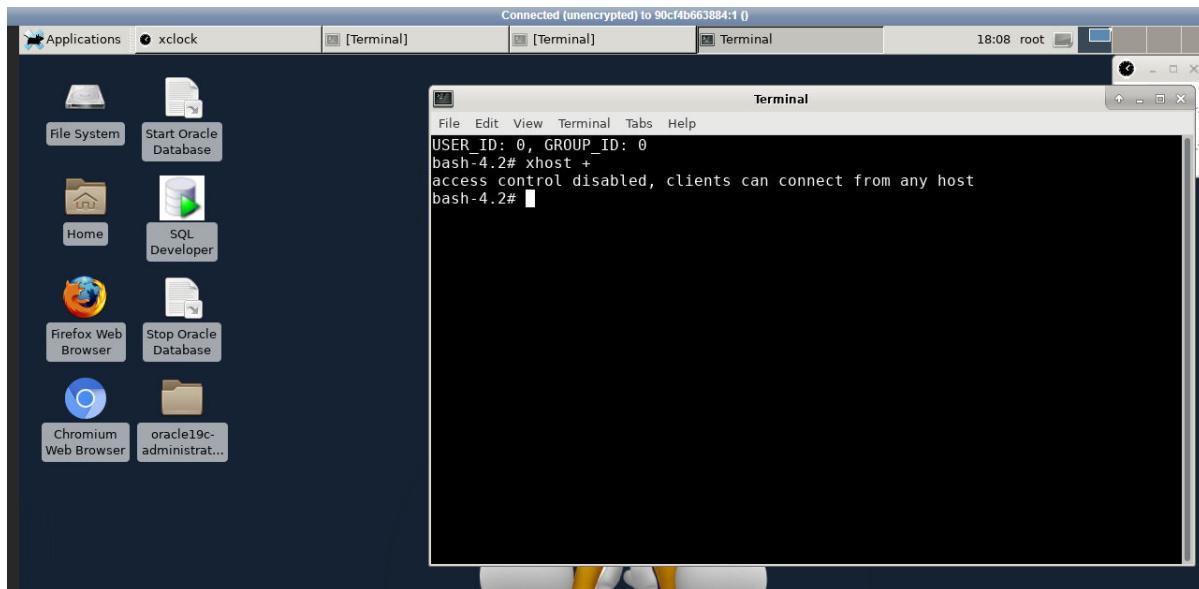
Practice 5-2: Switch to oracle user from terminal

Overview

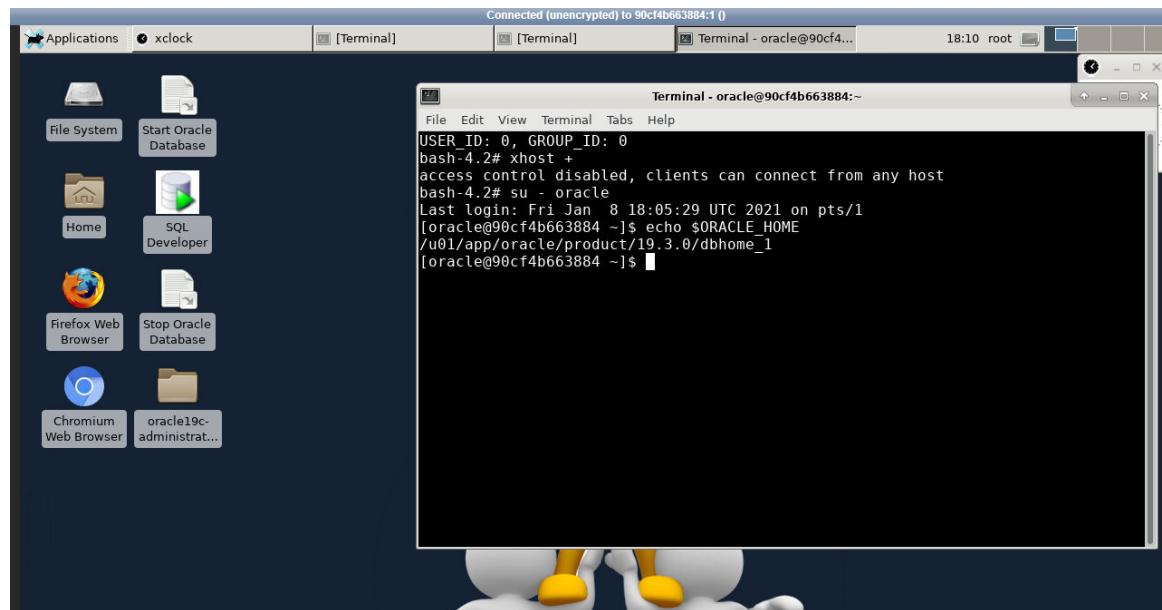
In this practice, you will switch to oracle user from terminal

Tasks

- Open terminal and run “xhost +” command as root user:



- 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.
 - a. 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 ~]$
```

- b. Source the `oraenv` script. `oraenv` sets the required environment variables needed for you to connect to your database instance. The `oraenv` script sets the `ORACLE_SID` and `ORACLE_HOME` environment variables and includes the `$ORACLE_HOME/bin` directory in the `PATH` environment variable setting. Environment variables that this

script sets will persist in the terminal window until you close it. For the ORACLE_SID value, enter FENAGODB.

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

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

```
[oracle@MYDBCS ~]$ set | grep ORACLE  
OLD_ORACLE_BASE=/u01/app/oracle  
ORACLE_BASE=/u01/app/oracle  
ORACLE_HOME=/u01/app/oracle/product/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.

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

```
[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>
```

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

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

3. View information about the containers in your CDB.

- a. Verify that you have a container database by querying the `V$DATABASE` view. The `NAME` column should contain `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.

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

NAME	CDB	CON_ID
FENAGODB	YES	0

```
SQL>
```

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

```
SQL> SHOW con_name
```

CON_NAME
CDB\$ROOT

```
SQL>
```

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

```
SQL> SHOW con_id
```

CON_ID
1

```
SQL>
```

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

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

BANNER
Oracle Database 19c EE High Perf Release 18.0.0.0.0 - Production

```
SQL>
```

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

```

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

NAME          CON_ID
-----  -----
CDB$ROOT      1
PDB$SEED      2
PDB1          3

SQL>

```

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

```

SQL> SHOW pdbs

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

SQL>

```

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

```

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

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

SQL>

```

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

```

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

```

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

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

12 rows selected.

SQL>

```

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

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

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

SQL>
```

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

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

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

SQL>
```

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

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

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

SQL>
```

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

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

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

USERNAME
-----
```

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

38 rows selected.

```
SQL>
```

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

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

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>
```

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

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

INSTANCE_NAME	STATUS	CON_ID
FENAGODB	OPEN	0

```
SQL>
```

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

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

CON_ID	NAME
1	SYS\$BACKGROUND
1	FENAGODB.588436052.oraclecloud.internal
1	FENAGODB.588436052.oraclecloud.internalXDB
1	SYS\$USERS
3	pdb1

```
SQL>
```

7. Exit SQL*Plus.

```
SQL > exit
Disconnected from Oracle Database 19c EE High Perf Release
18.0.0.0.0 - Production
Version 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.
 - a. Start SQL*Plus and connect to the root container as the `SYS` user with the `SYSDBA` privilege. Oracle allows any DBA group user at the operating system level to log into SQL*Plus without any authentication.

```
[oracle@MYDBCS ~]$ sqlplus / as sysdba  
...  
SQL>
```
 - b. Verify that `PDB1` is open. After DBCA creates a PDB, it opens it automatically. The results below indicate that the open mode is `READ WRITE`, which means `PDB1` is open. PDB users with the `SYSDBA`, `SYSOPER`, `SYSBACKUP`, `SYSDG`, `SYSKM`, or `SYSRAC` privilege can connect to a closed PDB; however, all other PDB users can connect only when the PDB is open.

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

```

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

SQL>

```

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

```

SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN;

Pluggable database altered.

SQL>

```

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

```

SQL> ALTER SESSION SET CONTAINER = PDB1;

Session altered.

SQL>

```

- e. Verify that the container name is PDB1.

```

SQL> SHOW con_name

CON_NAME
-----
PDB1
SQL>

```

2. Query the data dictionary to list the data files and temp files for PDB1.
- List the data files for PDB1 and the tablespaces to which they belong by querying the DBA_DATA_FILES view.

```

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

FILE_NAME                                TABLESPACE
-----
/u02/app/oracle/oradata/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      USERS
```

```
SQL>
```

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

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

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

```
SQL>
```

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

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

```
USERNAME
```

PDBADMIN
APEX_LISTENER
APEX_PUBLIC_USER
APEX_REST_PUBLIC_USER
FLOWS_FILES
APEX_050100
APEX_INSTANCE_ADMIN_USER
SCOTT

```
8 rows selected.
```

```
SQL>
```

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

- a. Disconnect from the PDB.

```
SQL > DISCONNECT
```

```
Disconnected from Oracle Database 19c EE High Perf Release  
18.0.0.0.0 - Production  
Version 19.3.0.0.0  
SQL>
```

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

```
SQL> SHOW user
```

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

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

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

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

4. Exit SQL*Plus.

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

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

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

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

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

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

4. Switch to PDB1.

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

5. Execute the `hr_main.sql` script and respond to the prompts as follows.

- Enter the password for the `HR` user as specified in the *Course Practice Environment: Security Credentials*.
- Enter `USERS` as the default tablespace for the `HR` user.
- Enter `TEMP` as the temporary tablespace for the `HR` user.
- 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 tablespace for HR as parameter 2:
Enter value for 2: USERS

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

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

PL/SQL procedure successfully completed.

User created.

User altered.

Grant succeeded.

...

Comment created.

Commit complete.

PL/SQL procedure successfully completed.

SQL>
```

e. Exit from SQL*Plus.

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

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

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

b. Query `USER_TABLES`.

```
SQL> SELECT table_name FROM user_tables;

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

7 rows selected.

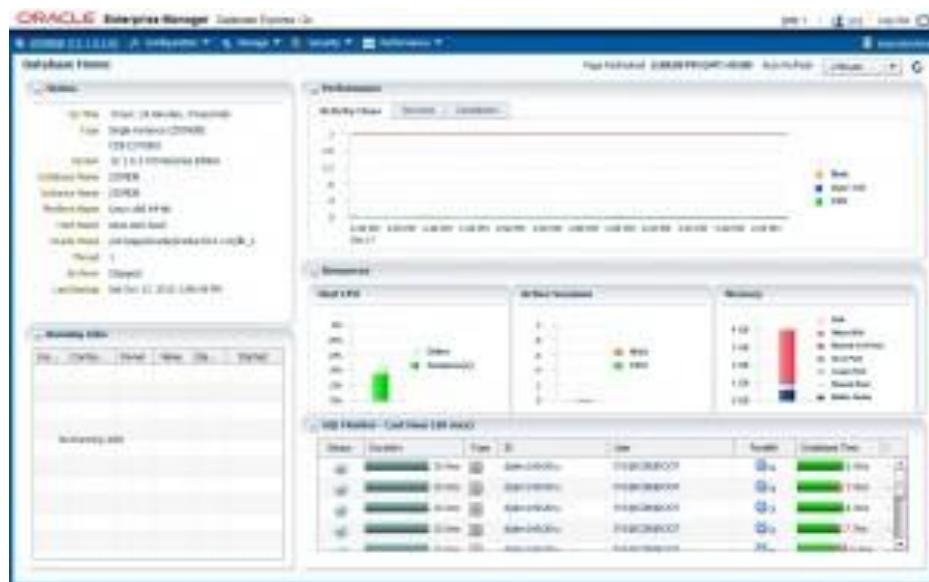
SQL>
```

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

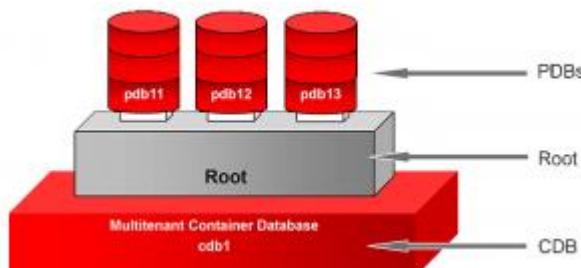
```
SQL> exit
Disconnected from Oracle Database 19c Enterprise Edition Release
18.0.0.0.0 - Production
Version 19.3.0.0.0
[oracle@MYDBCS human_resources]$ exit
```

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.



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**).



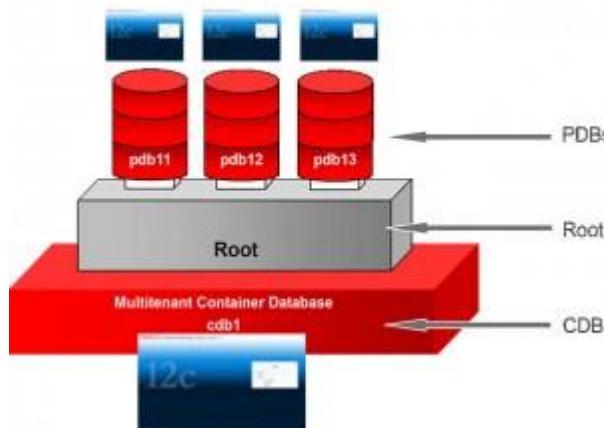
A **CDB** includes the following components:

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.



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

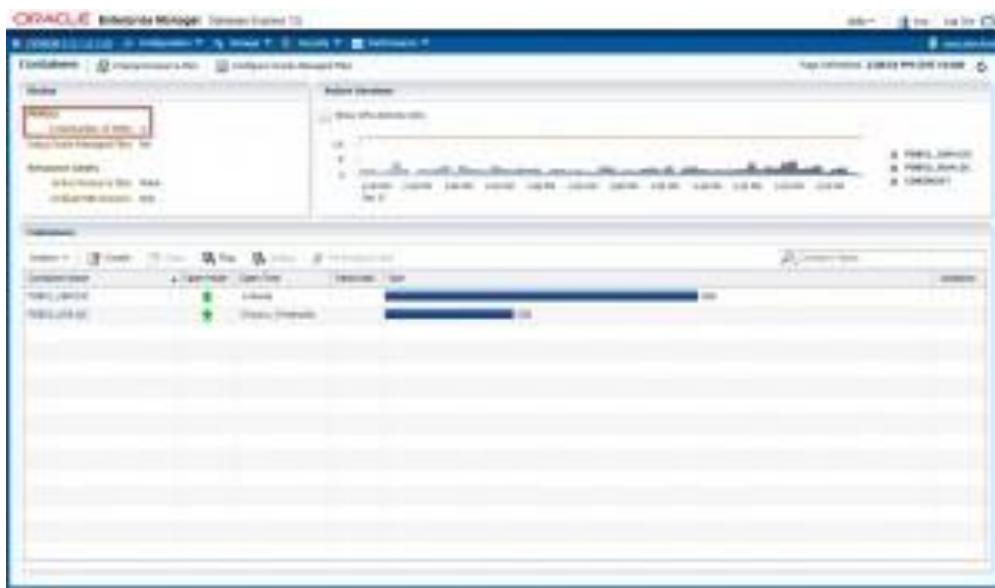
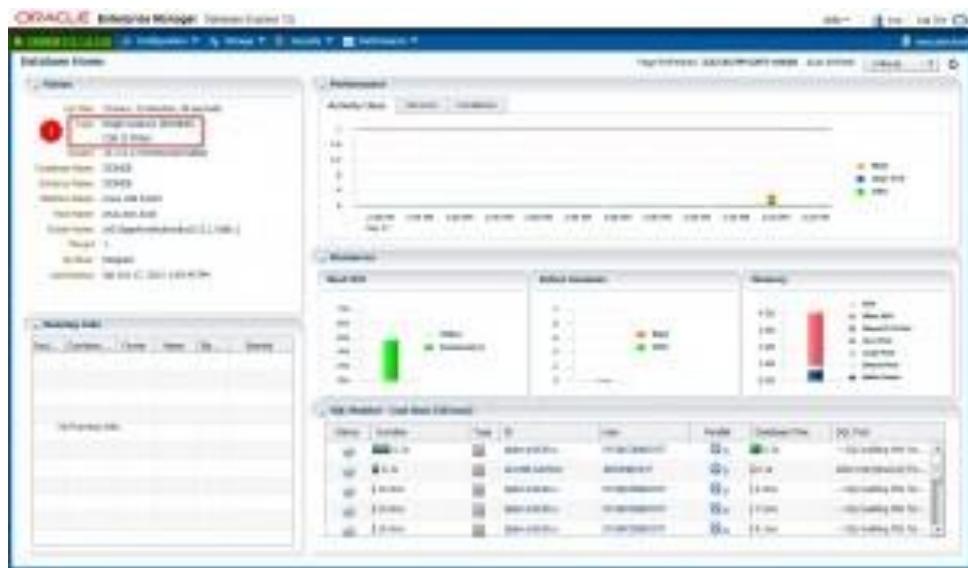
```
SQL> exec DBMS_XDB_CONFIG.SETHTTPSPORT(5500);
```

```
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



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

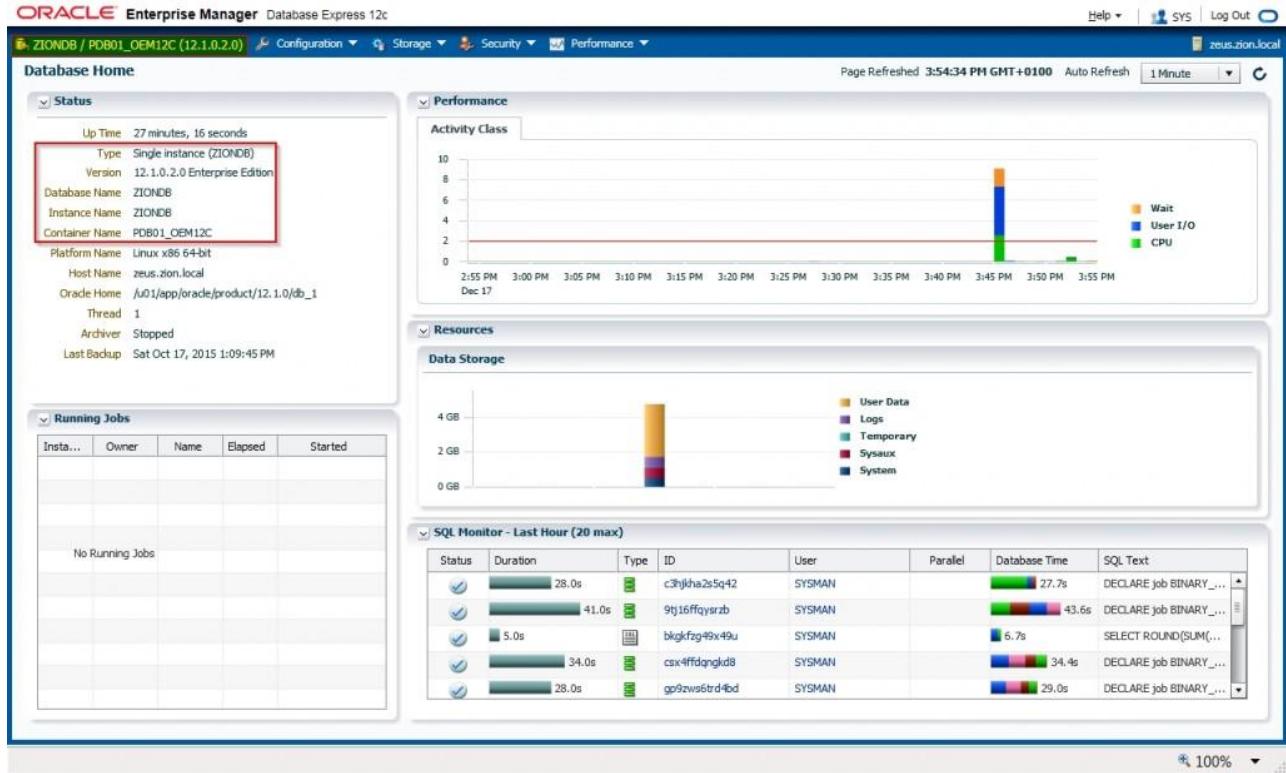
```
SQL> exec DBMS_XDB_CONFIG.SETHTTPSPORT(5501);
```

```
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**



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.

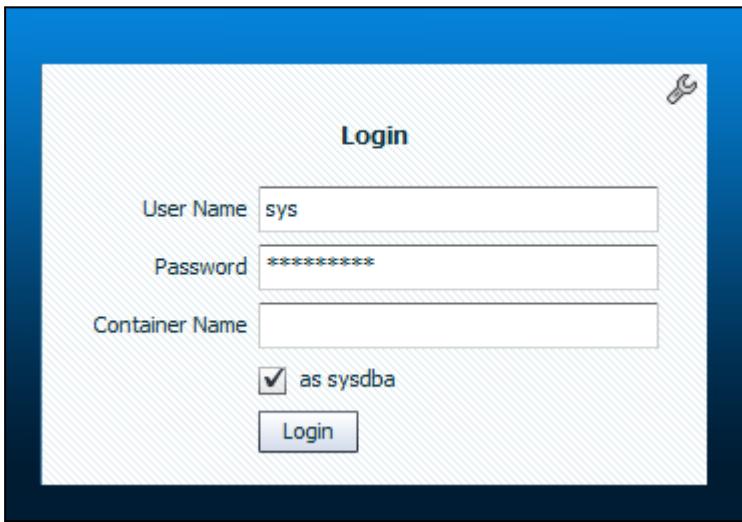
Assumptions

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:
 - a. Click **Advanced**.
 - b. Click **Add Exception**.
 - c. Click **Confirm Security Exception**.
3. On the Enterprise Manager Database Express login page, enter the following:
 - a. In the User Name field, enter `sys`.
 - b. In the Password field, enter the password you specified when you created the database deployment.
 - c. Leave the Container Name field empty.

d. Click **Login**.



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

Overview

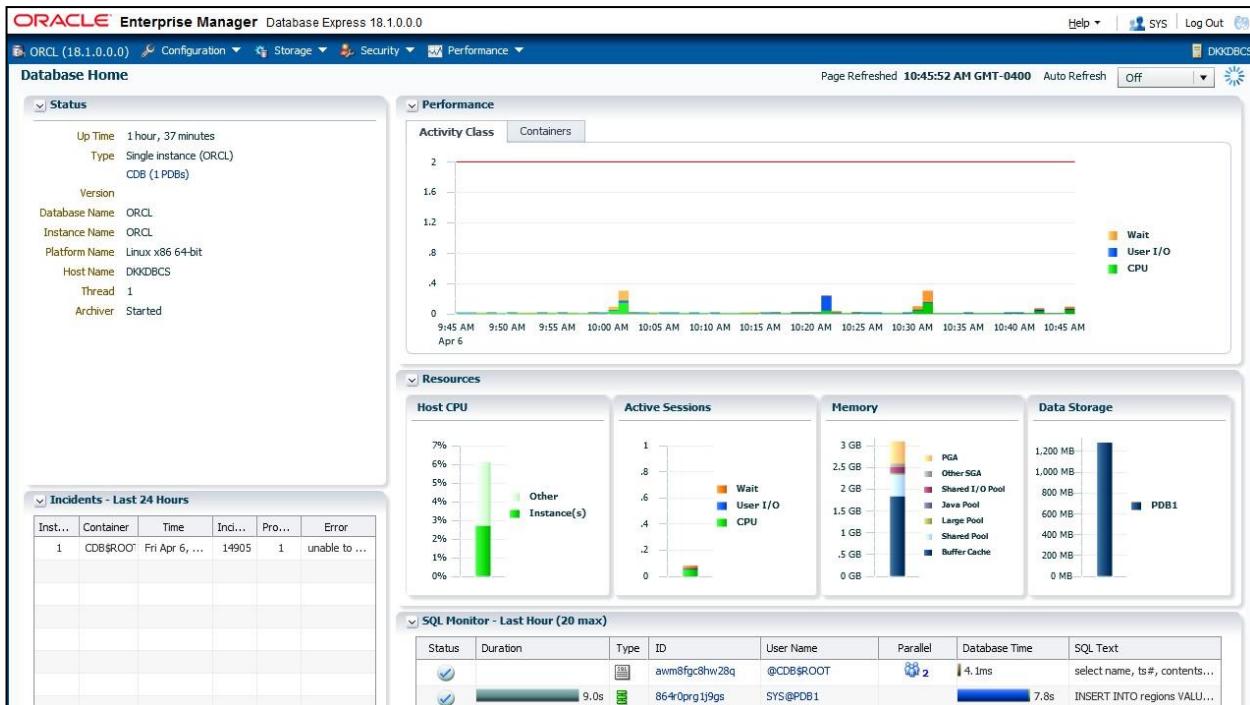
In this practice, you explore the differences in the Enterprise Manager Database Express interface when connected to a CDB and a PDB.

Assumptions

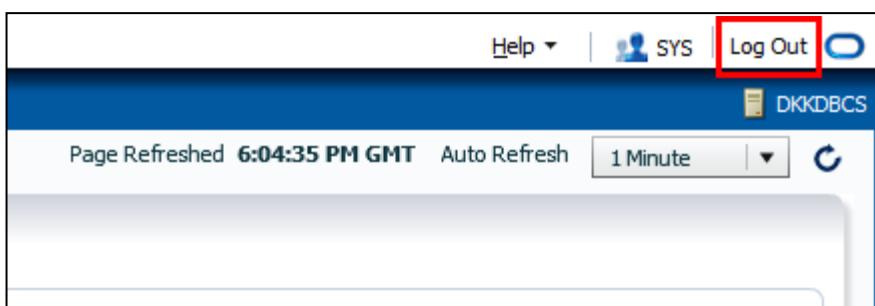
You logged in to Enterprise Manager Database Express as the `SYS` user in the previous practice.

Tasks

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



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



4. Now log in to EM Express to view information for PDB1.

- a. On the Login page for EM Express, in the User Name field enter **sys**.
- b. In the Password field, enter the password specified when you created the database deployment.
- c. Click **Login**.

The screenshot shows the 'Login' dialog box. It has fields for 'User Name' (sys), 'Password' (represented by asterisks), 'Container Name' (PDB1), and a checked checkbox labeled 'as sysdba'. A 'Login' button is at the bottom.

5. 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.

The screenshot shows the 'Database Home' page for the 'ORCL / PDB1 (18.1.0.0.0)' instance. The 'Container: PDB1' is highlighted with a red box. The 'Status' section displays the following information:

Parameter	Value
Up Time	1 hour, 42 minutes
Type	Single instance (ORCL)
Version	
Database Name	ORCL
Instance Name	ORCL
Container Name	PDB1
Platform Name	Linux x86 64-bit
Thread	1
Archiver	Started

6. Browse the Configuration, Storage, Security, and Performance menus.
7. When you are finished browsing through the menus, log out of Enterprise Manager Database Express.
8. 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 ~]$
```

2. Source the `oraenv` script.

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

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

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

4. Locate the default SPFILE for your database instance by using the `SHOW PARAMETER spfile` 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.

```
SQL> SHOW PARAMETER spfile  
  
NAME          TYPE  
-----  
spfile        string  
VALUE  
-----  
/u01/app/oracle/product/18.0.0/dbhome_1/dbs/spfileORCL.ora  
SQL>
```

5. View the `init.ora` file. This is the sample text initialization parameter file (PFILE) provided with the Oracle Database installation.

- a. Use the SQL*Plus `HOST` command to return to the operating system prompt.

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

- b. 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  
hc_ORCL.dat    init.ora          1kORCL      snapcf_ORCL.f  
hc_ORCL.dat    initORCL.ora     1kORCL      spfileORCL.ora  
initORCL.ora   initORCL.ora.ORG orapwORCL  tmp_PDB1.xml  
[oracle@MYDBCS dbs]$
```

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

- c. 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 1.25 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]$

```

- d. 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;
```

- e. Return to SQL*Plus.

```
[oracle@MYDBCS dbs]$ exit  
exit  
  
SQL>
```

6. 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.

- a. Shut down the database instance in IMMEDIATE mode.

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

- b. Use the HOST command to return to an operating system prompt.

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

- c. Change to the ORACLE_HOME/dbs directory.

```
[oracle@MYDBCS ~]$ cd $ORACLE_HOME/dbs  
[oracle@MYDBCS dbs]$
```

- d. 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]$
```

- e. Return to SQL*Plus.

```
[oracle@MYDBCS dbs]$ exit  
exit  
  
SQL>
```

- f. 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>

```

- g. 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.

```

SQL> SHOW PARAMETER spfile
NAME                           TYPE        VALUE
-----
spfile                         string
SQL>

```

7. Configure the database instance to once again start with the SPFILE.

- a. Shut down the database instance in IMMEDIATE mode.

```

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

```

- b. Use the HOST command to return to the operating system.

```

SQL> host
[oracle@MYDBCS ~]$

```

- c. Change to the ORACLE_HOME/dbs directory.

```

[oracle@MYDBCS ~]$ cd $ORACLE_HOME/dbs
[oracle@MYDBCS dbs]$

```

- d. Rename the orig_spfileORCL.orafile to spfileORCL.ora.

```

[oracle@MYDBCS dbs]$ mv orig_spfileORCL.ora spfileORCL.ora
[oracle@MYDBCS dbs]$

```

- e. Return to SQL*Plus.

```

[oracle@MYDBCS dbs]$ exit
exit

SQL>

```

- f. 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>
```

- g. Verify that the database instance was started with the SPFILE.

```
SQL> SHOW PARAMETER spfile
NAME                           TYPE        VALUE
-----
spfile                        string
VALUE
-----
/u01/app/oracle/product/18.0.0/dbhome_1/dbs/spfileORCL.ora
SQL>
```

8. 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>
```

2. View the values of the `DB_NAME` and `DB_DOMAIN` parameters. Together, these values create the global database name.
 - a. View the value of the `DB_NAME` parameter. This parameter specifies the current database identifier of up to eight characters. If you have multiple databases, the value of this parameter should match the Oracle instance identifier of each one to avoid confusion with other databases running on the system.

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

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

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

db_domain	string	588436052.oraclecloud.internal
SQL>		

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

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

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

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

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

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

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

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

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

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

SQL> SHOW PARAMETER sga		
NAME	TYPE	VALUE
-----	-----	-----
allow_group_access_to_sga	boolean	FALSE
lock_sga	boolean	FALSE
pre_page_sga	boolean	TRUE

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

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

SQL> SHOW PARAMETER undo_tablespace		
NAME	TYPE	VALUE
-----	-----	-----
undo_tablespace	string	UNDOTBS1
SQL>		

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

SQL> SHOW PARAMETER compatible		
NAME	TYPE	VALUE
-----	-----	-----
compatible	string	18.0.0
noncdb_compatible	boolean	FALSE
SQL>		

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

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

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

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

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

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

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

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

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

SQL> SHOW PARAMETER transactions		
NAME	TYPE	VALUE
-----	-----	-----
transactions	integer	519
transactions_per_rollback_segment	integer	5
SQL>		

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

SQL> SHOW PARAMETER db_files		
NAME	TYPE	VALUE
-----	-----	-----
db_files	integer	500
SQL>		

View Advanced Parameters

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

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

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

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

SQL> SHOW PARAMETER commit_wait		
NAME	TYPE	VALUE
-----	-----	-----
commit_wait	string	
SQL>		

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

```
SQL> SHOW PARAMETER shared_pool_size
```

NAME	TYPE	VALUE
shared_pool_size	big integer	0

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

```
SQL> SHOW PARAMETER db_block_size
```

NAME	TYPE	VALUE
db_block_size	integer	8192

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

```
SQL> SHOW PARAMETER db_cache_size
```

NAME	TYPE	VALUE
db_cache_size	big integer	0

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

```
SQL> SHOW PARAMETER undo_management
```

NAME	TYPE	VALUE
undo_management	string	AUTO

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

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

- View the `MEMORY_TARGET` parameter.

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

- View the `MEMORY_MAX_TARGET` parameter.

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

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

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

Query Views for Parameter Values

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

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

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

```

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

66 rows selected.

SQL>

```

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

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

Name	Null?	Type
NUM		NUMBER
NAME		VARCHAR2 (80)
TYPE		NUMBER
VALUE		VARCHAR2 (4000)
DISPLAY_VALUE		VARCHAR2 (4000)
DEFAULT_VALUE		VARCHAR2 (255)
ISDEFAULT		VARCHAR2 (9)
ISSES_MODIFIABLE		VARCHAR2 (5)
ISSYS_MODIFIABLE		VARCHAR2 (9)

```

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

SQL>

```

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

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

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

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

NAME	ISSYS_MOD	VALUE
lock_name_space	FALSE	
processes	FALSE	300
...		
multishard_query_data_consistency	IMMEDIATE	strong
multishard_query_partial_results	IMMEDIATE	not allowed

433 rows selected.

```
SQL>
```

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

```

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

```

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

9 rows selected.

SQL>

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

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

Name	Null?	Type
FAMILY		VARCHAR2 (80)
SID		VARCHAR2 (80)
NAME		VARCHAR2 (80)
TYPE		VARCHAR2 (11)
VALUE		VARCHAR2 (255)
DISPLAY_VALUE		VARCHAR2 (255)
ISSPECIFIED		VARCHAR2 (6)
ORDINAL		NUMBER
UPDATE_COMMENT		VARCHAR2 (255)
CON_ID		NUMBER

SQL>

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

NAME	VALUE

```

lock_name_space
processes          300
sessions
timed_statistics
timed_os_statistics
...
shrd_dupl_table_refresh_rate
multishard_query_data_consistency
multishard_query_partial_results

437 rows selected.

SQL>

```

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

```

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

SQL>

```

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

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

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

438 rows selected.

SQL>
```

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

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

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

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

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

NAME	VALUE
lock_name_space	
processes	300
sessions	472
timed_statistics	TRUE
timed_os_statistics	0
resource_limit	TRUE
...	
parallel_servers_target	16
common_user_prefix	
multishard_query_data_consistency	strong
multishard_query_partial_results	not allowed
457 rows selected.	
SQL>	

6. Exit SQL*Plus.

SQL> EXIT
Disconnected from Oracle Database 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>
```

2. Learn about the `NLS_DATE_FORMAT` parameter by querying the `V$PARAMETER` view.

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

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

NAME	ISSES	ISSYS_MOD	ISPDB	VALUE
nls_date_format	TRUE	FALSE	TRUE	

```
SQL>
```

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

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

NAME                      VALUE
-----
nls_territory              AMERICA

SQL>
```

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

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

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

SQL>
```

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

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

107 rows selected.

SQL>
```

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

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

```
Session altered.
```

```
SQL>
```

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

```
SQL> SELECT last_name, hire_date FROM hr.employees;
LAST_NAME          HIRE_DATE
-----
King                jun 17 2003
Kochhar              sep 21 2005
De Haan              jan 13 2001
Hunold               jan 03 2006
Ernst                may 21 2007
...
Mavris               jun 07 2002
Baer                 jun 07 2002
Higgins              jun 07 2002
Gietz                jun 07 2002

107 rows selected.
```

```
SQL>
```

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

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

8. Disconnect from PDB1 to end your session.

```
SQL> DISCONNECT
Disconnected from Oracle Database 18c Enterprise Edition Release
18.0.0.0.0 - Production
Version 18.1.0.0.0
SQL>
```

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

```

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

```

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

```

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

107 rows selected.

SQL>

```

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

```

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

```

Modify a Dynamic System-Level Parameter

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

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

```

SQL> exit

```

```

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

```

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

```

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

NAME          ISSES_ISSYS_MOD      VALUE
-----
job_queue_processes    FALSE IMMEDIATE    4000

SQL>

```

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

```

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

System altered.

SQL>

```

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

```

SQL> SHOW PARAMETER job

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

```

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

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

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

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

```
SQL> STARTUP
ORACLE instance started.

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

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

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

Modify a Static System-Level Parameter

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

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

```

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

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

SQL>

```

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

```

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

System altered.

SQL>

```

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

```

SQL> SHOW PARAMETER sec_max

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

```

4. Restart the database and then verify that the new value for the SEC_MAX_FAILED_LOGIN_ATTEMPTS parameter is updated.
 - a. Shut down the database instance with the IMMEDIATE mode.

```

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

SQL>

```

- ol style="list-style-type: none;">
 - b. Start the database instance by using the STARTUP command.

```

SQL> STARTUP
ORACLE instance started.


```

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

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

```
SQL> SHOW PARAMETER sec_max
NAME                           TYPE        VALUE
-----
sec_max_failed_login_attempts    integer      2
SQL>
```

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

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

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

SQL>
```

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

```
SQL> EXIT
```

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

6. On your Linux desktop, create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
 - a. Open a terminal window.
 - b. 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 ~]$
```

7. On your Linux desktop, open Firefox. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
8. On the Enterprise Manager Database Express login page, enter the following:
 - a. In the User Name field, enter **sys**.
 - b. In the Password field, enter the password you specified when you created the database deployment.
 - c. Leave the Container Name field empty.
 - d. Select **as sysdba**.
 - e. Click **Login**.

9. On the Home page, expand the **Configuration** menu and then select **Initialization Parameters**.

The screenshot shows the Oracle Enterprise Manager Database Express 18.1.0 interface. At the top, there's a navigation bar with tabs for Configuration, Storage, and Security. The Configuration tab is active and has a dropdown menu. The 'Initialization Parameters' option is highlighted with a mouse cursor. Below the navigation bar, the main area is titled 'Database Home' and shows various system status details. A 'Status' section includes fields like Up Time, Type, Version, Database Name, Instance Name, Platform Name, Host Name, Thread, and Archiver. To the right of the status section, a vertical sidebar lists options: Memory, Database Feature Usage, Current Database Properties, and Resource Management.

10. View the list of parameters by scrolling down.
11. 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.

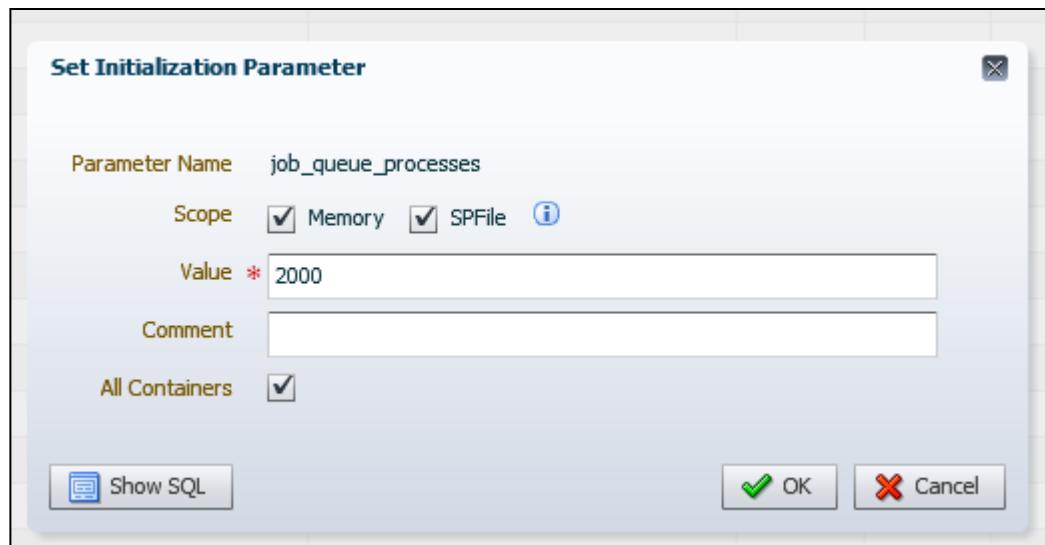
The screenshot shows a list of database parameters. At the top, there's a header bar with the text 'DKKDBCS' and a refresh icon. Below it, a message says 'Page Refreshed 8:05:01 PM GMT'. A search bar at the top contains the text 'job'. The main area is a table with columns labeled 'Basic', 'Basic', 'Type', and 'Description'. The first two columns are currently empty, while the third and fourth columns are visible.

12. The `JOB_QUEUE_PROCESSES` initialization parameter should be listed. Select `job_queue_processes` and click **Set**.

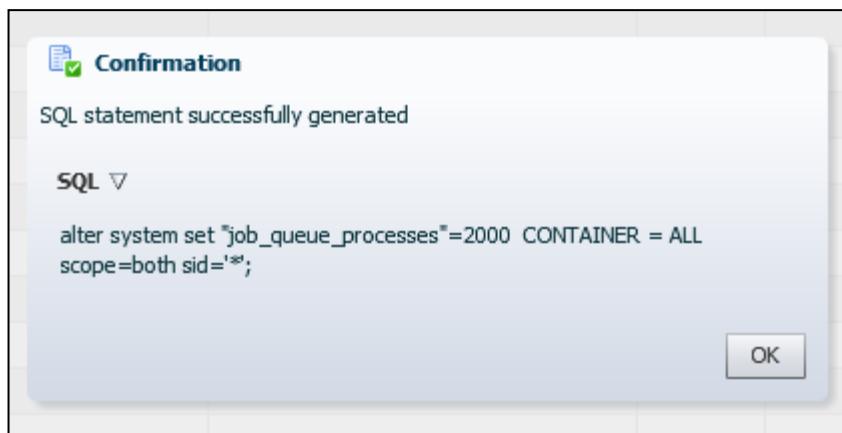
The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE® Enterprise Manager Database Express 18.1.0.0.0". The navigation bar includes tabs for Configuration, Storage, Security, and Performance. The main content area is titled "Initialization Parameters" and has two tabs: "Current" (selected) and "SPFile". A message states: "The parameter values listed here are currently used by open container(s)". Below this are buttons for View, Set..., Validate with SPA, and Help. A table lists initialization parameters:

Name	Set Parameter	Container Name	Value
max_datapump_jobs_per_pdb	*		100
job_queue_processes	*		4000

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



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



- a. 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.
- b. 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.
15. In the Set Initialization Parameter dialog box, click **OK**.
16. In the Confirmation dialog box, verify that the message states "Parameter job_queue_processes successfully updated," and click **OK**.
17. Notice that the job_queue_processes value is updated to 2000.

The screenshot shows the 'Initialization Parameters' page in Oracle Enterprise Manager. The top navigation bar includes 'ORACLE Enterprise Manager Database Express 18.1.0.0.0', 'ORCL (18.1.0.0.0)', 'Configuration', 'Storage', 'Security', and 'Performance'. The main title is 'Initialization Parameters'. Below it, there are two tabs: 'Current' (selected) and 'SPFile'. A message states: 'The parameter values listed here are currently used by open container(s.)'. Below this are buttons for 'View', 'Set...', 'Validate with SPA', and 'Help'. A table lists initialization parameters:

Name	Container Name	Value
max_datapump_jobs_per_pdb	*	100
job_queue_processes	*	2000

A red box highlights the 'Value' column for the 'job_queue_processes' row, which is '2000'.

18. 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 ~]$
```

2. Source the `oraenv` script.

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

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

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

4. 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>
```

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

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

6. 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>
```

7. 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>
```

8. 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>
```

9. 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>
```

10. Show the current container name.

```
SQL> SHOW con_name

CON_NAME
-----
CDB$ROOT

SQL>
```

11. Show the current user.

```
SQL> SHOW user

USER is "SYS"

SQL>
```

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

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

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

SQL>
```

13. 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.

```
SQL> SELECT trigger_name, trigger_type, triggering_event,
  2  trigger_body
  3  FROM dba_triggers
  4  WHERE trigger_name LIKE 'OPEN%';

TRIGGER_NAME      TRIGGER_TYPE      TRIGGERING_EVENT
-----
TRIGGER_BODY

OPEN_ALL_PLUGGABLES  AFTER EVENT          STARTUP
BEGIN
```

```
EXECUTE IMMEDIATE 'ALTER PLUGGABLE DATABASE ALL OPEN';
END;
SQL>
```

14. 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>
```

2. View the locations of the various diagnostics directories in the ADR. The results below have been formatted for easier reading.

- The path that corresponds to the Diag Alert entry in the NAME column is for the XML version. This path is /u01/app/oracle/diag/rdbms/orcl/ORCL/alert.
- The path that corresponds to the Diag Trace entry is for the text-only version. This path is /u01/app/oracle/diag/rdbms/orcl/ORCL/trace.

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

NAME	VALUE
Diag Enabled	TRUE
ADR Base	/u01/app/oracle
ADR Home	/u01/app/oracle/diag/rdbms/orcl/ORCL
Diag Trace	/u01/app/oracle/diag/rdbms/orcl/ORCL/trace
Diag Alert	/u01/app/oracle/diag/rdbms/orcl/ORCL/alert
Diag Incident	/u01/app/oracle/diag/rdbms/orcl/ORCL/incident
Diag Cdmp	/u01/app/oracle/diag/rdbms/orcl/ORCL/cdump
Health Monitor	/u01/app/oracle/diag/rdbms/orcl/ORCL/hm
Default Trace File	
	/u01/app/oracle/diag/rdbms/orcl/ORCL/trace/ORCL ora_2600.trc
Active Problem Count	1
Active Incident Count	6
11 rows selected.	
SQL>	

3. Exit SQL*Plus.

```
SQL> EXIT
```

Use an Editor to View the Alert Log

- View the XML version of the alert log. The `log.xml` file is the XML version of the alert log.
- 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]$
```

- 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]$
```

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

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

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

```

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

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

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

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

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

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

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

Use ADRCI to View the Alert Log

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

```
PATH=$PATH:$ORACLE_HOME/bin.
```

```
[oracle@MYDBCS trace]$ adrci

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>
```

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

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

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

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

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

4. Enter **?Starting ORACLE instance?** and press return. Press **N** to search from 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)
        4K           Configured            4          675844
NONE
        2048K                  0           1321            0
NONE

RECOMMENDATION:
 1. For optimal performance, configure system with expected
number
  of pages for every supported system pagesize prior to the next
instance restart operation.
*****
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
Initial number of CPU is 2
Number of processor cores in the system is 2
Number of processor sockets in the system is 1
search hit BOTTOM, continuing at TOP
```

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

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

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

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

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

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

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

Log DDL Statements in the DDL Log File

1. Determine if DDL logging is enabled in PDB1. If not, enable it by setting the value for the **ENABLE_DDL_LOGGING** initialization parameter to TRUE.
 - a. Start SQL*Plus and log in to the database as the **SYS** user with the **SYSDBA** privilege.

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

- b. Switch to PDB1.

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

SQL>
```

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

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

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

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

SQL>
```

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

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

3. Exit SQL*Plus.

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

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

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

5. List the contents of the log directory.

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

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

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

7. 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]$
```

8. 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 ~]$
```

2. Source the `oraenv` script.

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

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

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

4. View the initialization parameters used during dynamic service registration.

- a. `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.

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

- b. `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 PARAMETER service_names
NAME          TYPE    VALUE
-----
service_names  string  ORCL.588436052.oraclecloud.internal
SQL>
```

- c. 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.

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

- d. 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.

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

5. 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 ~]$
```

6. View the `tnsnames.ora` file.

- a. Change directories to `$ORACLE_HOME/network/admin`.

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

- b. List the files in this directory. The `tnsnames.ora` file is listed.

```
[oracle@MYDBCS admin]$ ls
listener.ora          samples      sqlnet.ora
```

```
listener.orapre_vnrcr_config shrept.lst tnsnames.ora  
[oracle@MYDBCS admin]$
```

- c. 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]$
```

7. 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.ora
# 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]$

```

8. 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>

```

9. View information about the default listener by using the Listener Control utility.

- a. View the operations that are available by using the help command.

```

LSNRCTL> help
The following operations are available
An asterisk (*) denotes a modifier or extended command:

start          stop          status         services
servacls       version       reload         save_config
trace          spawn         quit          exit
set*           show*
LSNRCTL>

```

- b. 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>
```

- c. 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.

- d. 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.

- e. 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>
```

- f. 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>
```

10. 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]$
```

2. Open `listener.ora` in an editor.

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

3. 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)  
    )  
  )
```

4. 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>
```

2. 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>
```

3. 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.ora
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>

```

4. 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>

```

2. 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>
```

3. 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.
 - a. If necessary, change the directory to `$ORACLE_HOME/network/admin`.

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

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

2. 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]$
```

3. Connect to PDB1 and verify the current container.

- a. 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.

- b. Verify that the current container name is PDB1.

```
SQL> SHOW con_name  
CON_NAME  
-----  
PDB1
```

```
SQL>
```

4. 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 ~]$
```

2. Source the oraenv script.

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

3. Start SQL*Plus and connect to the root container as the SYS user with the SYSDBA privilege. This method of connecting uses OS authentication.

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

4. 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>
```

5. 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>
```

6. 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.

7. 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>
```

2. Show the current user by issuing the `SHOW USER` command. You are not connected as any user.

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

3. Connect to the root container as `c##CDB_ADMIN1` and exercise the `SYSDBA` privilege.

```
SQL> CONNECT c##CDB_ADMIN1/password AS SYSDBA  
Connected.  
SQL>
```

4. Show the current container name.

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

5. 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>
```

6. 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>
```

7. Disconnect from the root container.

```
SQL> DISCONNECT
...
SQL>
```

8. 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>
```

9. 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>
```

10. 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>
```

11. Switch to PDB1 by issuing the ALTER SESSION command.

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

SQL>
```

12. 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>
```

2. 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>
```

3. 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>
```

4. Disconnect c##CDB_ADMIN1 from PDB1.

```
SQL> DISCONNECT  
...  
SQL>
```

5. Connect to PDB1 as PDB1_ADMIN1.

```
SQL> CONNECT PDB1_ADMIN1/password@PDB1  
Connected.  
SQL>
```

6. Show the current user. You are connected as PDB1_ADMIN1.

```
SQL> SHOW USER  
USER is "PDB1_ADMIN1"  
SQL>
```

7. 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>
```

8. 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>
```

9. 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>
```

2. 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>
```

3. Grant the CREATE SESSION privilege to INVENTORY.

```
SQL> GRANT CREATE SESSION TO INVENTORY;  
  
Grant succeeded.  
  
SQL>
```

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

```
SQL> SELECT DISTINCT username FROM dba_users WHERE common='NO'  
ORDER BY username;  
  
USERNAME  
-----  
APEX_050100  
APEX_INSTANCE_ADMIN_USER
```

```
APEX_LISTENER  
APEX_PUBLIC_USER  
APEX_REST_PUBLIC_USER  
FLOWS_FILES  
HR  
INVENTORY  
PDB1_ADMIN1  
PDBADMIN  
SCOTT  
  
11 rows selected.
```

```
SQL>
```

Connect as INVENTORY and Verify Privileges

1. Disconnect PDB1_ADMIN1 from PDB1.

```
SQL> DISCONNECT  
...  
SQL>
```

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

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

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

```
SQL> SELECT * FROM session_privs ORDER BY privilege;  
  
PRIVILEGE  
-----  
CREATE SESSION  
  
SQL>
```

4. Exit SQL*Plus.

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

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>
```

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

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

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

```
SQL> col granted_role format a10  
SQL> SELECT granted_role, admin_option FROM cdb_role_privs WHERE  
grantee='PDBADMIN';  
  
GRANTED_RO ADM  
----- ---  
PDB_DBA YES
```

```
DBA          NO  
SQL>
```

4. List the system privileges granted to the `PDB_DBA` role by querying the `ROLE_SYS_PRIVS` view.

- a. Switch to `PDB1`. You must be connected to `PDB1` to retrieve data, and you must be connected as the `SYS` user.

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

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

```
SQL> SELECT privilege FROM role_sys_privs WHERE role='PDB_DBA'  
ORDER BY privilege;  
  
PRIVILEGE  
-----  
CREATE PLUGGABLE DATABASE  
CREATE SESSION  
SET CONTAINER  
  
SQL>
```

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

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

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

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

```
PRIVILEGE
```

```
-----  
CREATE SESSION  
SET CONTAINER
```

```
SQL>
```

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

Grant the DBA Role to PDBADMIN

1. Grant the DBA role locally to PDBADMIN.

```
SQL> GRANT dba TO pdbadmin;
```

```
Grant succeeded.
```

```
SQL>
```

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

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

```
GRANTED_RO
```

```
-----
```

```
DBA
```

```
PDB_DBA
```

```
SQL>
```

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

```
SQL> EXIT
```

```
...
```

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

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

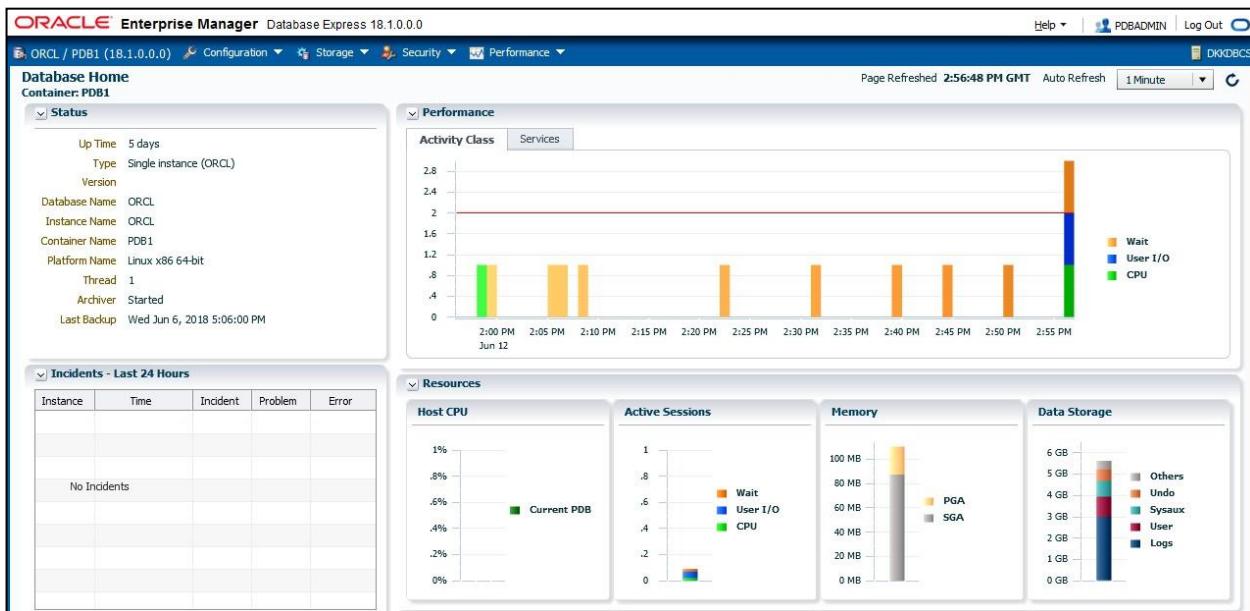
Log in to Enterprise Manager Database Express (PDB1)

1. Create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.
 - a. Open a terminal window.
 - b. 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 ~]$
```

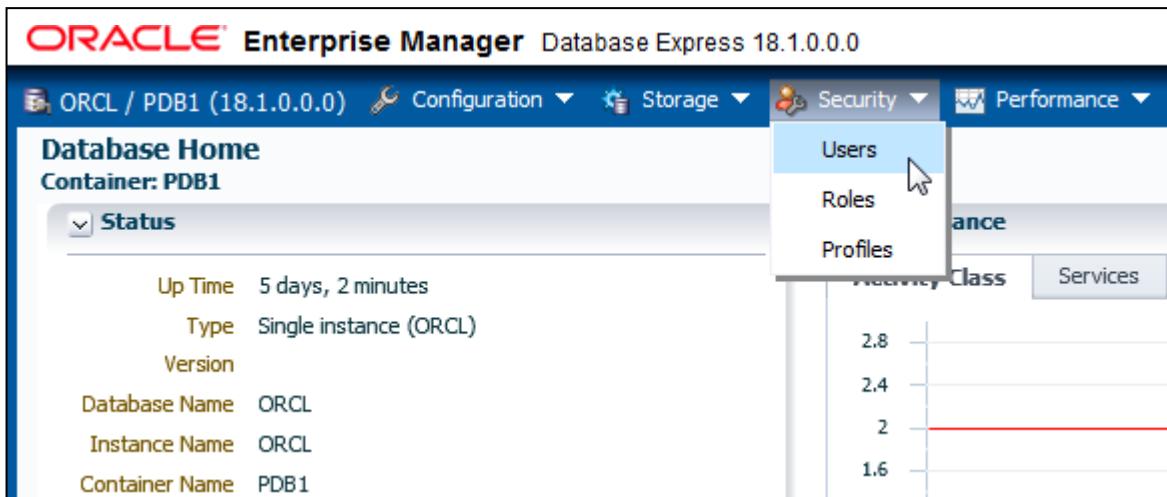
2. Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
3. On the Enterprise Manager Database Express login page, enter the following:
 - a. In the User Name field, enter PDBADMIN.
 - b. In the Password field, enter the password you specified when you created the database deployment.
 - c. In the Container Name field, enter PDB1.

- Click **Login**.
- View the Database Home page, which shows information for PDB1.



View Privileges and Roles for PDBADMIN

- On the Security menu, select **Users**.



- Scroll down the list of users and click **PDBADMIN**.

3. The View User: PDBADMIN page is displayed. On the **Privileges and Roles** tab, click **DBA**.

View User: PDBADMIN
Container: PDB1

Account Summary

Name	PDBADMIN
Profile	DEFAULT
Authentication	PASSWORD
Common	No
Expiration Date	Sun Nov 25, 2018 6:46:28 PM
Default Tablespace	SYSTEM
Temporary Tablespace	TEMP
Account Status	OPEN
Created	Tue May 29, 2018 6:46:28 PM

Details

Privileges & Roles	Object Privileges	Quotas
Privileges & Roles <small>i</small>		
<small>Edit</small> <small>Revoke</small>		
Privilege	With Admin	
DBA	<input checked="" type="checkbox"/>	
PDB_DBA	<input checked="" type="checkbox"/>	
UNLIMITED TABLESPACE		

4. 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.
5. Use the search box to look for specific privileges and roles, for example, **CREATE PROFILE**.

Create a Local Profile

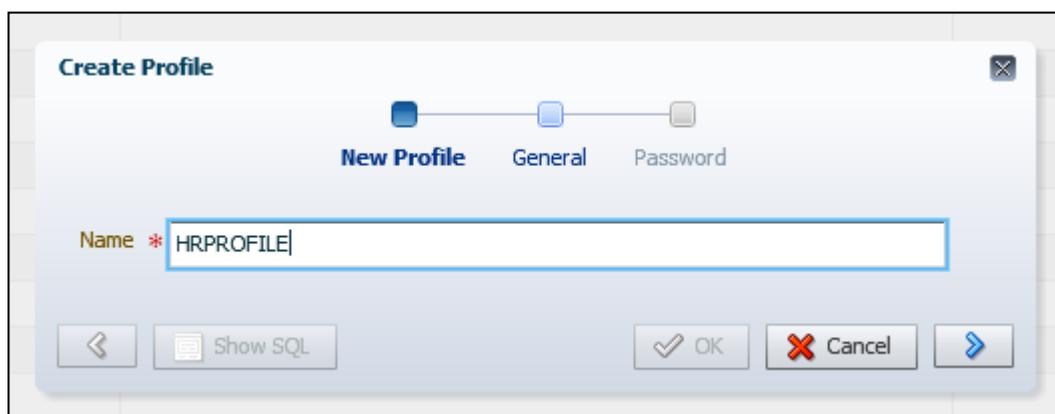
1. On the **Security** menu, select **Profiles**.

2. Click **Create Profile**.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE Enterprise Manager Database Express 18.1.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation is a section titled "Profiles" with "Container: PDB1". A toolbar below this has "Actions" dropdown, "Create Profile" (highlighted with a blue arrow), and "Drop Profile". The main area displays a table of profiles:

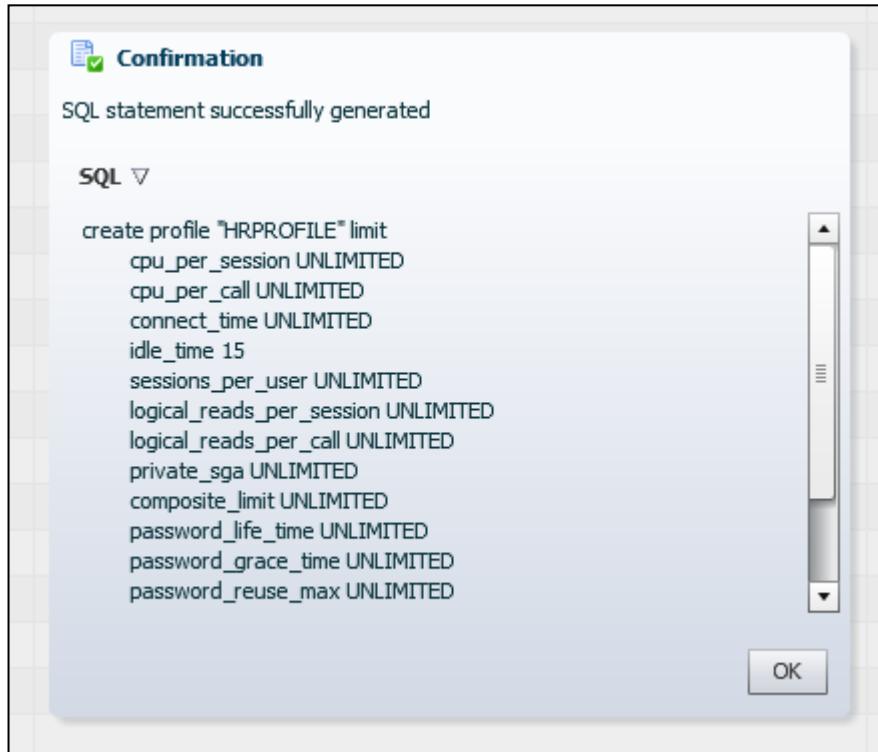
Profile	Connect Time (Min.)
DBAASSECURE	DEFAULT
DEFAULT	UNLIMITED
ORA_STIG_PROFILE	DEFAULT

3. The Create Profile wizard is displayed. On the New Profile page, in the Name box, enter **HRPROFILE** and click **Next** (blue arrow).



4. 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.
5. On the Password page, review the password options. All should be set to default values of Unlimited or Null.

6. Click **Show SQL** to review the SQL command for this task. Click **OK** to close the window.



7. You are returned to the Password page. Click **OK** to close it.
8. On the Confirmation page, note the message "SQL statement has been processed successfully" and click **OK**.
9. Verify that `HRPROFILE` is in the list of profiles.

The screenshot shows the Oracle Enterprise Manager Database Express interface, specifically the "Profiles" page for the "PDB1" container. The top navigation bar includes links for Configuration, Storage, Security, and Performance. The main content area displays a table of profiles:

Profile	Connect Time (Min.)
DBAASSECURE	DEFAULT
DEFAULT	UNLIMITED
HRPROFILE	UNLIMITED
ORA_STIG_PROFILE	DEFAULT

Set the `RESOURCE_LIMIT` Initialization Parameter

1. On the Configuration menu, select **Initialization Parameters**.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The top navigation bar includes tabs for Configuration, Storage, Security, and Performance. The left sidebar lists profiles: DBAASSECURE, DEFAULT, HRPROFILE, and ORA_STIG_PROFILE. The main content area displays initialization parameters for the DEFAULT profile. A context menu is open over the 'Resource Management' row, with options like Memory, Database Feature Usage, Current Database Properties, and Resource Management.

	Connect Time (Min.)
DEFAULT	UNLIMITED
UNLIMITED	UNLIMITED
DEFAULT	DEFAULT

2. In the **Name** search field, enter `resource_limit`.
3. The `RESOURCE_LIMIT` initialization parameter is listed in the table. Verify that the `RESOURCE_LIMIT` value is set to `true`.

The screenshot shows the Initialization Parameters page for the PDB1 container. The table lists the `resource_limit` parameter under the Resource Manager section, with its value set to `true`.

Name	Value
resource_limit	true

4. If `RESOURCE_LIMIT` is not set to `true`, perform the following steps:
 - a. Click **Cancel** to return to the table.
 - b. Select `resource_limit` and click **Set**.
 - c. In the Set Initialization Parameter dialog box, select the `true` option and click **OK**.
 - d. In the Confirmation dialog box, click **OK**.
5. 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 ~]$
```

2. Source the `oraenv` script.

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

3. 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>
```

4. 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>
```

5. 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.

```
SQL> COL limit FORMAT A20  
SQL> SELECT resource_type, resource_name, limit FROM  
dba_profiles WHERE profile='HRPROFILE';  
  
RESOURCE RESOURCE_NAME LIMIT  
----- ----- -----  
KERNEL COMPOSITE_LIMIT UNLIMITED  
KERNEL SESSIONS_PER_USER UNLIMITED
```

```

KERNEL    CPU_PER_SESSION          UNLIMITED
KERNEL    CPU_PER_CALL            UNLIMITED
KERNEL    LOGICAL_READS_PER_SESSION UNLIMITED
KERNEL    LOGICAL_READS_PER_CALL   UNLIMITED
KERNEL    IDLE_TIME                15
...
PASSWORD  PASSWORD_LOCK_TIME     UNLIMITED
PASSWORD  PASSWORD_GRACE_TIME    UNLIMITED
PASSWORD  INACTIVE_ACCOUNT_TIME  DEFAULT

17 rows selected.

SQL>

```

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.

- 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> ! oerr ora 2377
02377, 00000, "invalid profile limit %s"
//  *Cause: A value of 0 or lower was specified for the limit.
//  *Action: Specify a limit greater than 0. For password
profile parameters,
//           some additional restrictions apply:
//           * For the INACTIVE_ACCOUNT_TIME profile
parameter, the specified
//           limit cannot be less than 15 days.
//           * For the PASSWORD_GRACE_TIME profile parameter,
0 is allowed
//           as a permissible value.

SQL>

```

- 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>

```

8. Query the DBA_PROFILES view again to confirm that the limit is set.

```
SQL> SELECT resource_type, resource_name, limit FROM
  dba_profiles WHERE profile = 'HRPROFILE';
```

RESOURCE	RESOURCE_NAME	LIMIT
KERNEL	COMPOSITE_LIMIT	UNLIMITED
KERNEL	SESSIONS_PER_USER	UNLIMITED
KERNEL	CPU_PER_SESSION	UNLIMITED
...		
PASSWORD	PASSWORD_LOCK_TIME	UNLIMITED
PASSWORD	PASSWORD_GRACE_TIME	UNLIMITED
PASSWORD	INACTIVE_ACCOUNT_TIME	15

17 rows selected.

```
SQL>
```

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

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

10. 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 PDBADMIN 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.
 - a. Open a terminal window.
 - b. 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 ~]$
```
2. Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
3. On the Enterprise Manager Database Express login page, enter the following:
 - a. In the User Name field, enter PDBADMIN.
 - b. In the Password field, enter the password you specified when you created the database deployment.
 - c. In the Container Name field, enter PDB1.
4. Click Login.

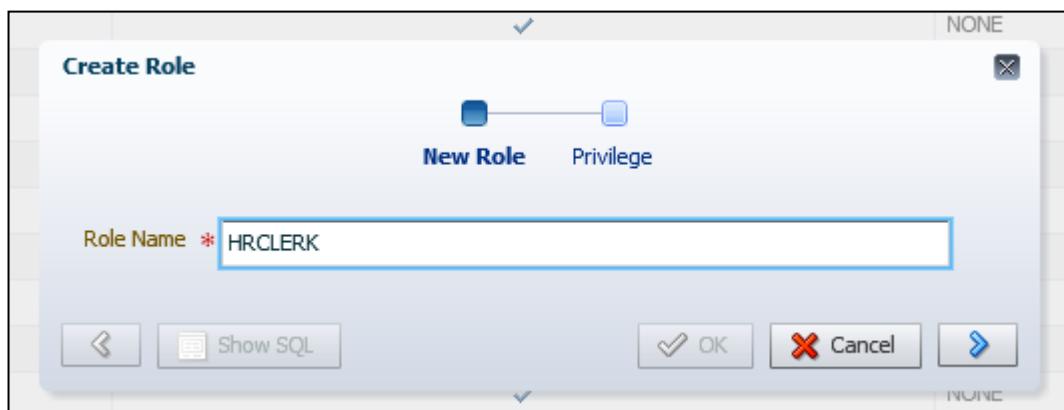
Create the HRCLERK Role

1. Select Security and then Roles.

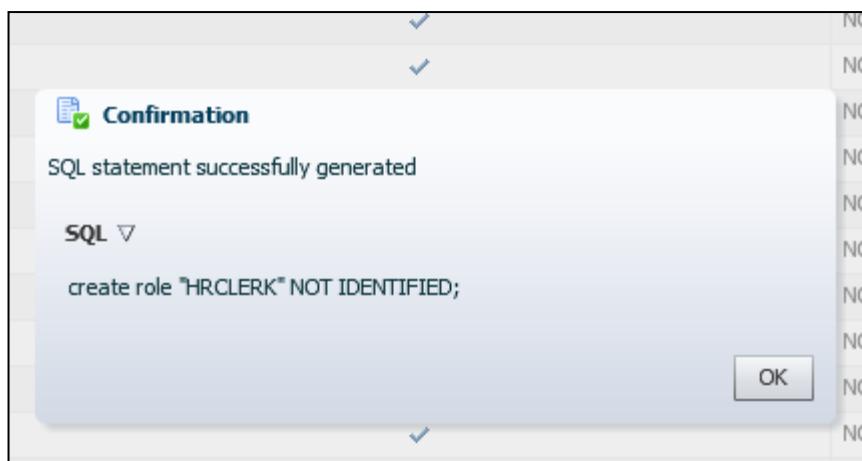
2. The roles are listed in a table. Click **Create Role**.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE® Enterprise Manager Database Express 18.1.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation is a toolbar with a "Create Role" button, which is highlighted with a mouse cursor. The main content area is titled "Roles" and "Container: PDB1". It features a table with columns for Actions, Role Name, and Privileges. The table contains four rows: "ADM_PARALLEL_EXECUTE_TASK", "APEX_ADMINISTRATOR_READ_ROLE", and "APEX_ADMINISTRATOR_ROLE".

3. The Create Role Wizard is displayed. On the New Role page, in the Role Name box, enter **HRCLERK** and click Next.



4. 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.
5. After reviewing the SQL statement, click **OK** to return to the Privilege page.



6. On the Privilege page, click **OK**.
7. In the Confirmation dialog box, click **OK**.

- Verify that the **HRCLERK** role is listed in the table.
- In the table, select the **HRCLERK** role.

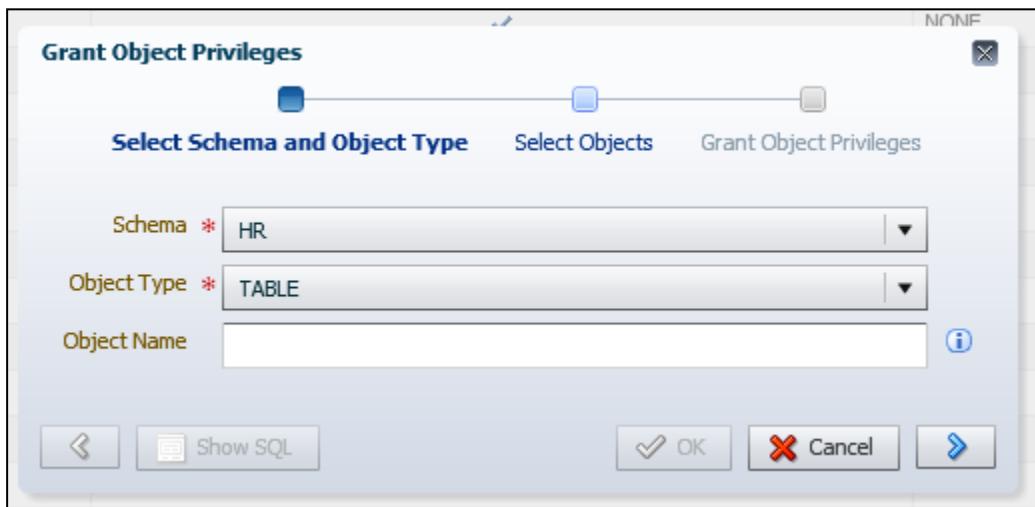
The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE Enterprise Manager Database Express 18.1.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. The main content area is titled "Roles" and shows "Container: PDB1". Below this, there is a table with two rows: "Actions" and "Role Name". Under "Role Name", three roles are listed: "HRCLERK", "HS_ADMIN_EXECUTE_ROLE", and "HS_ADMIN_ROLE".

- Select Actions and then Grant Object Privileges.

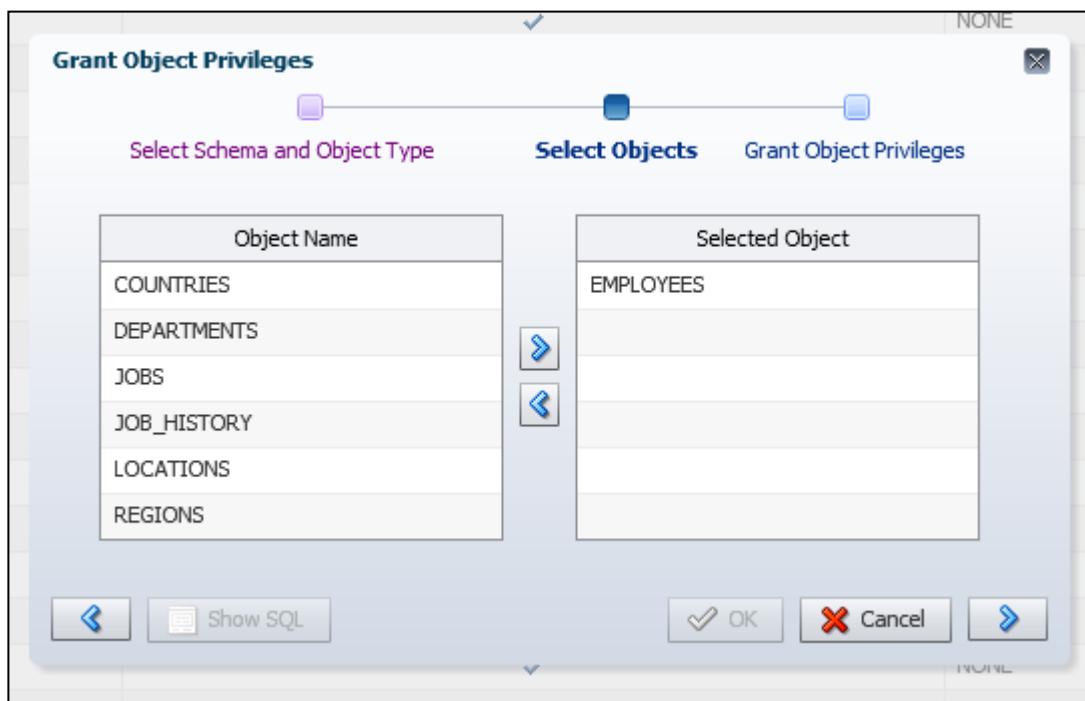
This screenshot shows the same Oracle Enterprise Manager interface as the previous one, but with a context menu open over the "HRCLERK" role in the list. The menu options are: Create Role, Drop Role, View Details, Alter Privileges & Roles, and Grant Object Privileges. The "Grant Object Privileges" option is highlighted with a blue selection bar and has a cursor arrow pointing towards it.

- The Grant Object Privileges Wizard is displayed. On the Select Schema and Object Type page, do the following and click the arrow (Next).
 - In the Schema drop-down list, select **HR**.
 - In the Object Type drop-down list, ensure that **TABLE** is selected.

- c. Leave the Object Name box blank.



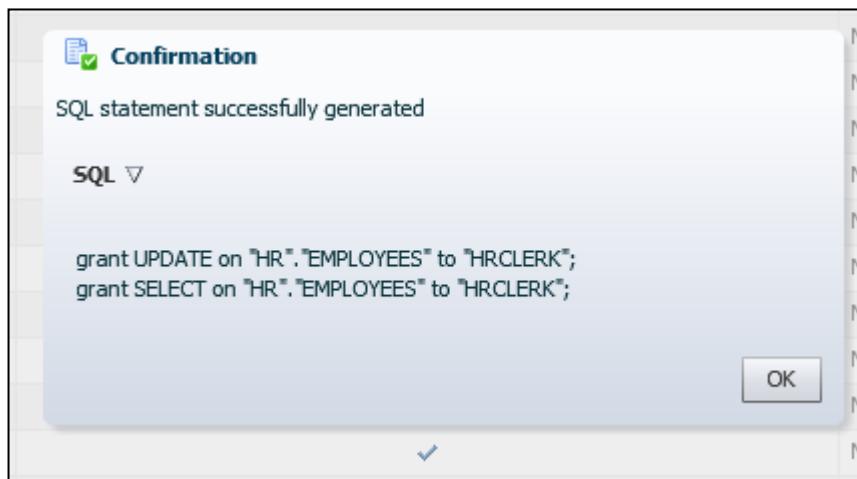
12. On the Select Objects page, select **EMPLOYEES** in the Object Name column and click the right arrow button to move it to the Selected Object column. Click the arrow (next).



13. On the Grant Object Privileges page, select the check boxes for the **SELECT** and **UPDATE** privileges.



14. Click **Show SQL**.
15. The Confirmation dialog box displays the SQL commands that are generated to perform the grant. Click **OK**.



16. On the Grant Object Privileges page, click **OK** to complete the grant.
17. 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.

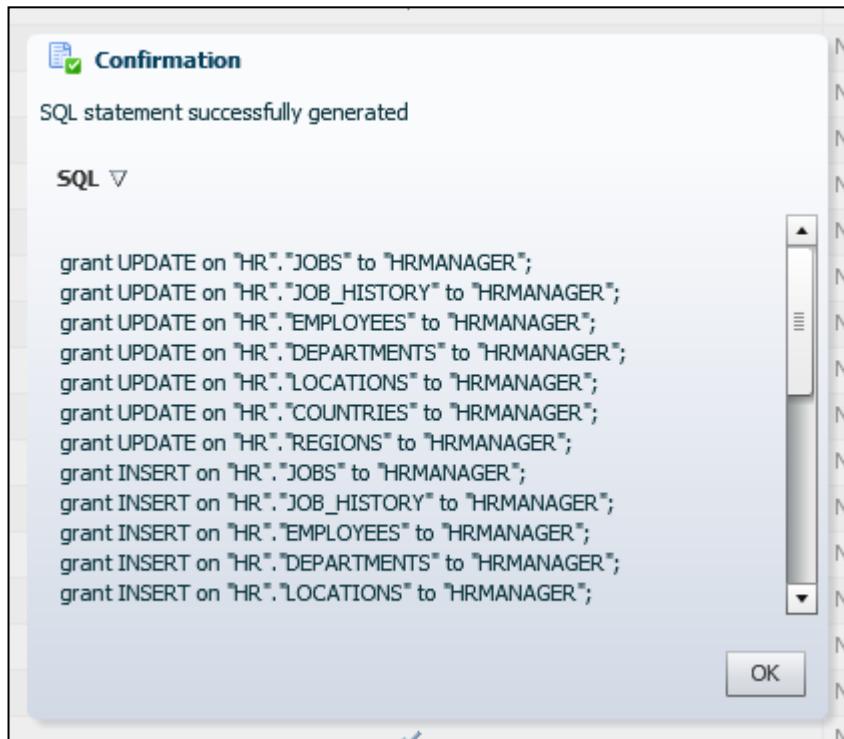
1. If you are not on the Roles page, select **Security** and then **Roles**. The roles are listed in a table.
2. Click **Create Role**.
3. The Create Role Wizard is displayed. On the New Role page, in the Role Name box, enter **HRMANAGER** and click the arrow (next).
4. On the Privilege page, click **OK**.
5. In the Confirmation dialog box, click **OK**.
6. Verify that the **HRMANAGER** role is listed in the table.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE Enterprise Manager Database Express 18.1.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation bar, the page title is "Roles" and the container is "PDB1". A toolbar below the title contains "Actions" (dropdown), "Create Role" (button), and "Drop Role" (button). The main content area is a table with a single column for "Role Name". The table rows contain the following values:

Role Name
HRMANAGER
HS_ADMIN_EXECUTE_ROLE
HS_ADMIN_ROLE
HS_ADMIN_SELECT_ROLE

7. In the table, select the **HRMANAGER** role. Select **Actions** and then **Grant Object Privileges**.
8. The Grant Object Privileges Wizard is displayed. On the Select Schema and Object Type page, do the following and then click the arrow (next).
 - a. In the Schema drop-down list, select **HR**.
 - b. In the Object Type drop-down list, select **TABLE**.
 - c. Leave the Object Name box blank.
9. 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).
10. On the Grant Object Privileges page, select the check boxes for the **DELETE**, **INSERT**, **SELECT**, and **UPDATE** privileges.

11. Click **Show SQL**.



12. The Confirmation dialog box displays the SQL commands that are generated to perform the grant. Click **OK**.
13. On the Grant Object Privileges page, click **OK** to complete the grant.
14. In the Confirmation dialog box, click **OK**. You have created the `HRMANAGER` role and granted the required privileges.
15. 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 manager	CREATE SESSION privilege HRCLERK role HRMANAGER role	EM Express
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.

- Open a terminal window.
- 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 ~]$
```

- Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
- On the Enterprise Manager Database Express login page, enter the following:
 - In the User Name field, enter **PDBADMIN**.
 - In the Password field, enter the password you specified when you created the database deployment.
 - In the Container Name field, enter **PDB1**.
- 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.

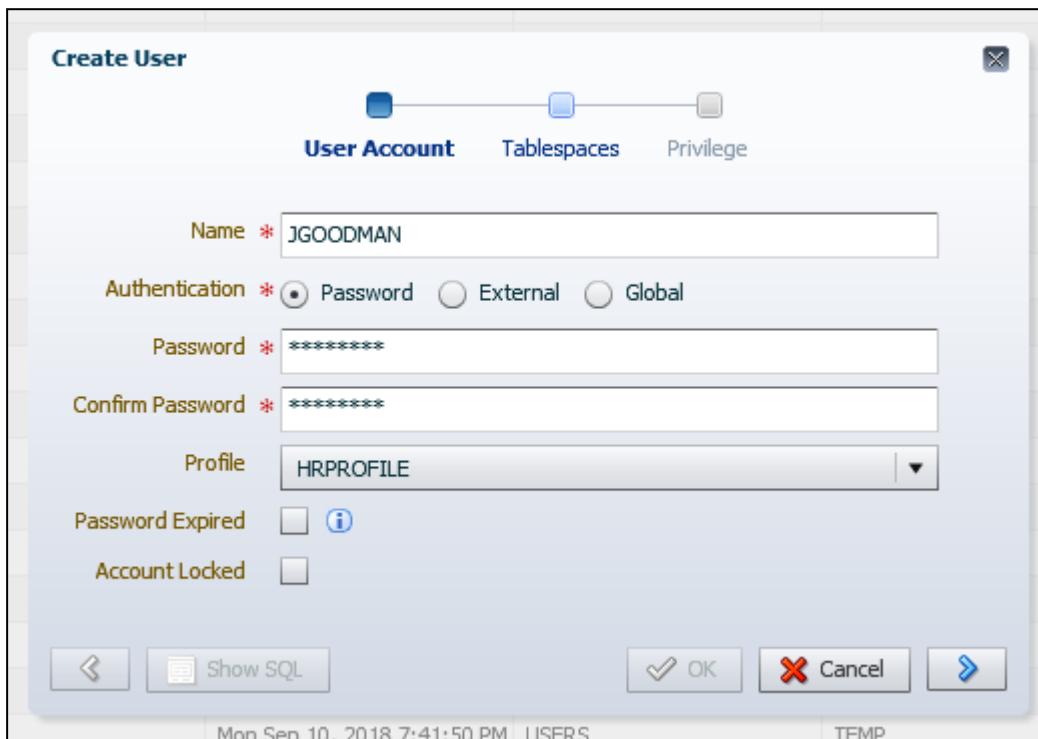
- Select **Security** and then **Users**.
- Click **Create User**.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE Enterprise Manager Database Express 18.1.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation is a sub-menu for "Container: PDB1". The main content area is titled "Users" and shows a table of existing users. The table has columns for Name, Account Status, Common, and Expiration Date. Two users are listed: "ANONYMOUS" and "APEX_050100". The "Create User" button is highlighted with a blue border and a cursor icon pointing at it. Other buttons in the row include "Create Like" and "Drop User".

Name	Account Status	Common	Expiration Date
ANONYMOUS			Wed Feb 7, 2018 9:45:00 AM
APEX_050100			Mon Feb 19, 2018 7:45:00 AM

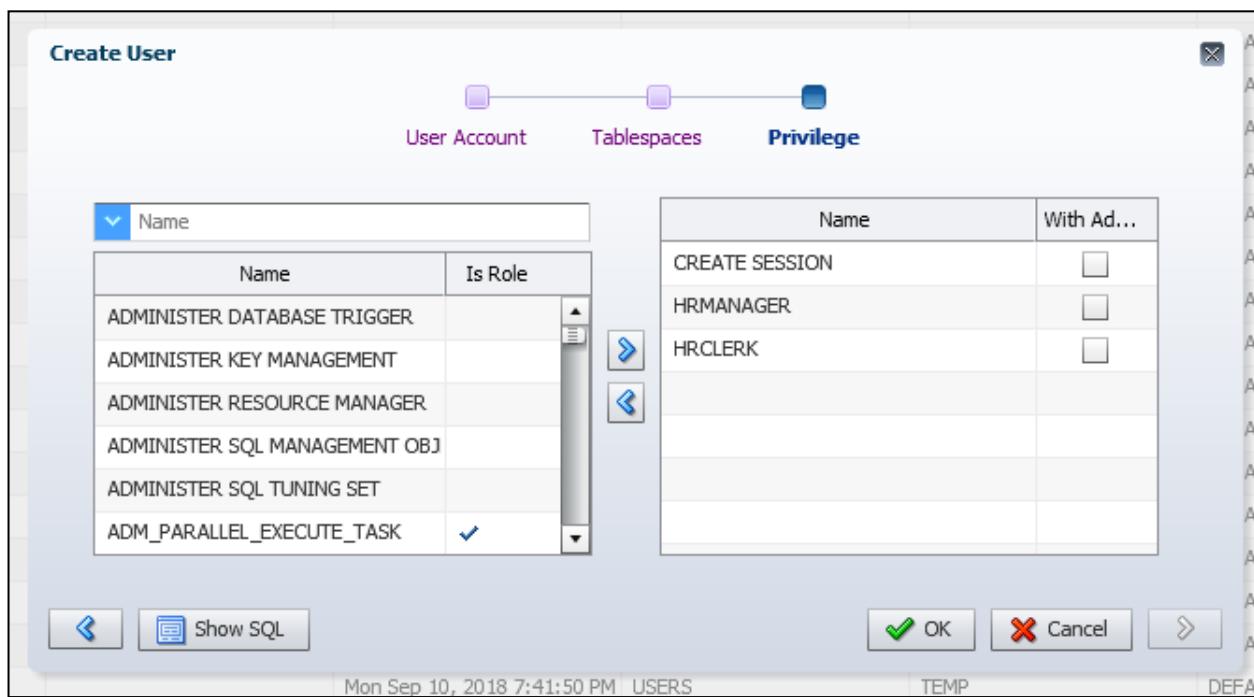
- The Create User Wizard is displayed. On the User Account page, enter or select the following values and then click Next.
 - Name: Enter **JGOODMAN**
 - Authentication: Password
 - Password and Confirm Password: Refer to *Course Practice Environment: Security Credentials* for the password value.
 - Profile: Select **HRPROFILE**.

- Leave the check boxes for Password Expired and Account Locked deselected.



4. On the Tablespaces page, ensure that the following tablespaces are selected and then click Next.
 - Default Tablespace: **USERS**
 - Temporary Tablespace: **TEMP**

- 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.



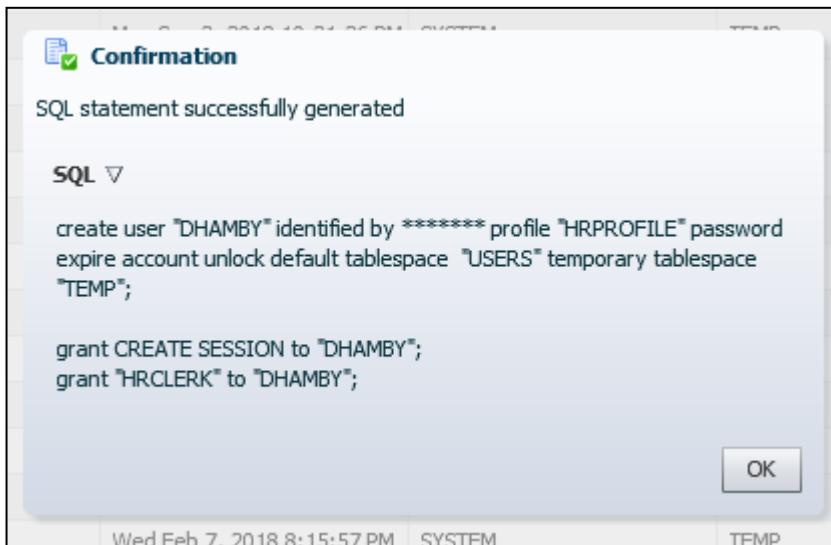
- Click **Show SQL**.
- After reviewing the SQL statement, click **OK**.
Note: In Practice 9-7 Configuring a Default Role for a User, you will assign the **HRCLERK** role to be this user account's default role.
- Click **OK** to execute the SQL statement.
- In the Confirmation dialog box, click **OK**.
- 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.

- If you are not on the Users page, expand **Security** and then select **Users**.
- Click **Create User**.
- 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.



8. 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.

- Copy to the clipboard the SQL statements in the previous step.
- Minimize, but don't close, EM Express.
- On the Linux desktop, select **Applications**, then **Accessories**, and then **gedit Text Editor**.

- d. Select **Edit** and then **Paste** to paste the SQL statements.
- e. Change every occurrence of **DHAMBY** to **&&username**. There are three occurrences.
- f. Change the one occurrence of **HRCLERK** to **&&role**.
- g. Replace the asterisks with the actual password. Refer to *Course Practice Environment: Security Credentials* for the password value.
- h. 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;
```

- i. Select **File** and then **Save As**.
 - j. Browse to **/home/oracle/labs**. In the Name box, enter **CreateHRUser.sql**. Click **Save**. The file is saved and formatted.
 - k. Select **File** and then **Quit**.
9. Return to EM Express, click **OK** to close the Confirmation dialog box, and click **OK** to finish creating the user.
 10. In the Confirmation dialog box, click **OK**.
 11. 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.
 12. 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]$
```

3. 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 \
```

```
> oracle@ your_compute_node_IP_Address:/home/oracle/CreateHRUser.sql  
CreateHRUser.sql  
[oracle@edvm .ssh] $
```

4. 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 ~]$
```

5. Source the `oraenv` script.

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

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

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

7. 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.

```
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.  
  
old    1: grant CREATE SESSION to &&username  
new    1: grant CREATE SESSION to RPANDYA  
  
Grant succeeded.  
  
Enter value for role: HRCLERK  
old    1: grant &&role to &&username  
new    1: grant HRCLERK to RPANDYA  
  
Grant succeeded.  
  
SQL>
```

8. Return to EM Express and click the Reload current page button to refresh the data.
9. Scroll down the list and verify that the user `RPANDYA` has been created as expected.
10. 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>
```

3. 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>
```

4. 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>
```

- 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.

- 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>
```

- 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>
```

- 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>
```

- Roll back the delete operation because this was just a test.

```
SQL> ROLLBACK;  
Rollback complete.  
  
SQL>
```

- Confirm that you still have 107 rows in the HR.EMPLOYEES table.

```
SQL> SELECT COUNT(*) FROM hr.employees;  
  
COUNT (*)
```

```
-----
```

```
107
```

```
SQL>
```

6. Disconnect from PDB1.

```
SQL> DISCONNECT
```

```
...
```

```
SQL>
```

Test the Idle Time Limit in HRP PROFILE

If you recall, in Practice 9-4 Creating a Local Profile in EM Express and Locking Accounts, you created a profile named `HR PROFILE`. 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>
```

2. Wait for 15 minutes. You can leave this terminal window open while waiting.
3. 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 `HR PROFILE` 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>
```

4. 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>
```

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

```
SQL> COLUMN granted_role FORMAT A20  
SQL> SELECT granted_role, default_role FROM dba_role_privs WHERE  
grantee='JGOODMAN';  
  
GRANTED_ROLE          DEF  
-----  ---  
HRCLERK              YES  
HRMANAGER             YES  
  
SQL>
```

3. 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>
```

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

```
SQL> SELECT granted_role, default_role FROM dba_role_privs WHERE  
grantee='JGOODMAN';  
  
GRANTED_ROLE      DEF  
-----  
HRCLERK          YES  
HRMANAGER        NO  
  
SQL>
```

5. Disconnect PDBADMIN from PDB1.

```
SQL> DISCONNECT  
...  
SQL>
```

Enable a Non-Default Role

1. Connect to PDB1 as JGOODMAN.

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

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

```
SQL> SELECT * FROM session_roles;  
  
ROLE  
-----  
HRCLERK  
  
SQL>
```

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

```
SQL> SET ROLE HRMANAGER;
```

```
Role set.
```

```
SQL>
```

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

```
SQL> SELECT * FROM session_roles;
```

```
ROLE
```

```
-----
```

```
HRMANAGER
```

```
SQL>
```

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

```
SQL> SET ROLE HRMANAGER, HRCLERK;
```

```
Role set.
```

```
SQL>
```

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

```
SQL> SELECT * FROM session_roles;
```

```
ROLE
```

```
-----
```

```
HRCLERK
```

```
HRMANAGER
```

```
SQL>
```

7. Exit SQL*Plus.

```
SQL> EXIT
```

```
...
```

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

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 ~]$
```

2. 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 ~]$
```

3. Find out more about the `oracle` user. For example, verify that `oracle` is part of the `dba` group.
 - a. 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 ~]$
```

- b. 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 ~]$
```

- c. 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>
```

2. View the columns in the `V$PWFILE_USERS` view by issuing the `DESCRIBE` command.

```
SQL> DESCRIBE v$pwfile_users
```

Name	Null?	Type
USERNAME		VARCHAR2 (128)

SYSDBA	VARCHAR2 (5)
SYSOPER	VARCHAR2 (5)
SYSASM	VARCHAR2 (5)
SYSBACKUP	VARCHAR2 (5)
SYSDG	VARCHAR2 (5)
SYSKM	VARCHAR2 (5)
ACCOUNT_STATUS	VARCHAR2 (30)
PASSWORD_PROFILE	VARCHAR2 (128)
LAST_LOGIN	TIMESTAMP (9) WITH TIME ZONE
LOCK_DATE	DATE
EXPIRY_DATE	DATE
EXTERNAL_NAME	VARCHAR2 (1024)
AUTHENTICATION_TYPE	VARCHAR2 (8)
COMMON	VARCHAR2 (3)
CON_ID	NUMBER

SQL>

3. 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>
```

4. 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).

```
SQL> SELECT account_status, sysdba from v$pwfile_users WHERE
username='SYS';

ACCOUNT_STATUS          SYSDBA
-----
OPEN                  TRUE

SQL>
```

5. 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 ~]$
```

2. Source the `oraenv` script.

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

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

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

SQL>

```

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

```

SQL> ALTER PLUGGABLE DATABASE PDB2 OPEN;

```

```
Pluggable database altered.
```

```
SQL>
```

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

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

CON_ID	NAME	OPEN_MODE
2	PDB\$SEED	READ ONLY
3	PDB1	READ WRITE
4	PDB2	READ WRITE

```
SQL>
```

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

```
SQL> !lsnrctl status
```

```
LSNRCTL for Linux: Version 18.0.0.0.0 - Production on 22-MAR-2018 16:00:59
```

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

```
Connecting to
```

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=MYDBCS.compute-588436052.oraclecloud.internal) (PORT=1521)))
```

```
STATUS of the LISTENER
```

Alias	LISTENER
Version	TNSLSNR for Linux: Version 18.0.0.0.0
- Production	
Start Date	19-MAR-2018 15:23:07
Uptime	3 days 0 hr. 37 min. 52 sec
Trace Level	off
Security	ON: Local OS Authentication
SNMP	OFF
Listener Parameter File	
/u01/app/oracle/product/18.0.0/dbhome_1/network/admin/listener.ora	
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=tcp) (HOST=MYDBCS.compute-
588436052.oraclecloud.internal) (PORT=5500)) (Security=(my_wallet_
directory=/u01/app/oracle/admin/ORCL/xdb_wallet)) (Presentation=H
TTP) (Session=RAW))

Services Summary...

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

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

Service
"68035adc452a1241e0532618120ad62d.588436052.oraclecloud.internal"
" has 1 instance(s).

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

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

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

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

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

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

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

Service "PDB2.588436052.oraclecloud.internal" has 1 instance(s).

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

The command completed successfully

SQL>

```

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

```

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

SQL>

```

Note: Alternatively, you could have switched to PDB2 by using the ALTERSESSION command.

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

8. Explore PDB2.

- a. Show the current container.

```
SQL> SHOW con_name
```

```
CON_NAME
```

```
-----  
PDB2
```

```
SQL>
```

- b. Show the current container ID.

```
SQL> SHOW con_id
```

```
CON_ID
```

```
-----  
4
```

```
SQL>
```

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

```
SQL> COLUMN name FORMAT A20
```

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

```
NAME
```

```
-----  
PDB2
```

```
SQL>
```

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

```
SQL> col file_name format a55
```

```
SQL> col tablespace_name format a10
```

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

```
FILE_NAME
```

```
TABLESPACE
```

```
-----  
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D SYSTEM
```

```
0A3C0D/datafile/o1_mf_system_fk2t2tmq_.dbf
```

```
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D SYSAUX
```

```
0A3C0D/datafile/o1_mf_sysaux_fk2t2tmv_.dbf
```

```
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D UNDOTBS1
```

```
0A3C0D/datafile/o1_mf_undotbs1_fk2t2tmy_.dbf
```

```
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D USERS
```

```
0A3C0D/datafile/o1_mf_users_fk2t2tn4_.dbf
```

```
SQL>
```

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

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

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

SQL>
```

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

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

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

7 rows selected.

SQL>
```

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

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

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

```
XS$NULL

39 rows selected.

SQL>
```

h. Exit SQL*Plus.

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

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

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

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

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

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

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

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

[oracle@MYDBCS admin] $
```

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

```
PDB2 =
  (DESCRIPTION =
```

```

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

```

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

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

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

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

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

[oracle@MYDBCS admin]$
```

10. Connect to PDB2 by using the new service name and verify the current container.
 - a. Start SQL*Plus and connect to PDB2 as the SYSTEM user by using the PDB2 net service name.

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

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

```
SQL> SHOW con_name  
  
CON_NAME  
-----  
PDB2  
SQL>
```

- c. Exit SQL*Plus.

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

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 `oracleuser`.

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 ~]$
```

2. Source the `oraenv` script.

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

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

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

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

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

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

```
NAME
```

```
-----  
/u02/app/oracle/oradata/ORCL/system01.dbf  
/u02/app/oracle/oradata/ORCL/sysaux01.dbf  
/u02/app/oracle/oradata/ORCL/undotbs01.dbf  
/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf  
/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf  
/u02/app/oracle/oradata/ORCL/users01.dbf  
/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf  
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf  
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf  
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf  
/u02/app/oracle/oradata/ORCL/PDB1/PDB1_users01.dbf  
/u02/app/oracle/oradata/ORCL/659622D851BF1AE2E0533620C40AD6D5/da  
tafile/o1_mf_users_fjv8c711_.dbf  
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da  
tafile/o1_mf_system_fk2t2tmq_.dbf  
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da  
tafile/o1_mf_sysaux_fk2t2tmv_.dbf  
/u02/app/oracle/oradata/ORCL/6D975E8B80B85F14E0537A051D0A3C0D/da  
tafile/o1_mf_undotbs1_fk2t2tmy_.dbf
```

```
16 rows selected.
```

```
SQL>
```

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

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

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

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

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

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

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

1. Switch to `PDB1`.

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

```
Session altered.
```

```
SQL>
```

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

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

USERNAME          ACCOUNT_STATUS
-----
HR                OPEN

SQL>
```

- 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.

- 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 ~]$
```

- Source the `oraenv` script.

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

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

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

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

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

      SALARY
-----
     24000

SQL>
```

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

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

107 rows updated.

SQL>
```

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

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

      SALARY
-----
     26400

SQL>
```

Window 1: Clone PDB1 as PDB3

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

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

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

```
SQL> CREATE PLUGGABLE DATABASE PDB3 FROM PDB1
  2  CREATE _FILE _DEST='/u02/app/oracle/oradata/ORCL/PDB3'
  3  KEYSTORE IDENTIFIED BY keystore_password;

Pluggable database created.

SQL>
```

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

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

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

SQL>
```

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

```
SQL> ALTER PLUGGABLE DATABASE PDB3 OPEN;

Pluggable database altered.

SQL>
```

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

```
SQL> SHOW PDBS

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

SQL>
```

Window 2: Commit the Transaction

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

```
SQL> COMMIT;

Commit complete.

SQL>
```

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

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

```
SALARY
-----
26400

SQL>
```

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

Answer: Continue to the next section to find out.

Window 1: Explore PDB3

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

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

Session altered.

SQL>
```

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

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

SALARY
-----
24000

SQL>
```

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

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

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

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

NAME
-----
PDB3

SQL>
```

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

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

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

SQL>
```

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

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

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

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

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

```
SQL>
```

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

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

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

14 rows selected.

SQL>
```

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

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

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

```
39 rows selected.  
  
SQL>
```

10. Exit SQL*Plus.

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

Window 2: Return Salary Values to Their Original Values

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

```
SQL> UPDATE employees SET salary=salary / 1.1;  
  
107 rows updated.  
  
SQL>
```

3. Commit the transaction.

```
SQL> COMMIT;  
Commit complete.  
  
SQL>
```

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

```
SQL> SELECT salary FROM hr.employees WHERE employee_id = 100;  
  
SALARY  
-----  
24000  
  
SQL>
```

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

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

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.

- a. Start SQL*Plus and log in to ORCL with the `SYSDBA` privilege.

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

- b. Close PDB3.

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

```
SQL> ALTER PLUGGABLE DATABASE PDB3 CLOSE IMMEDIATE;  
  
Pluggable database altered.  
  
SQL>
```

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

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

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

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

```
SQL> ALTER PLUGGABLE DATABASE PDB3  
2 UNPLUG INTO '/u02/app/oracle/oradata/PDB3.xml'  
3 ENCRYPT USING TransPDB3;  
  
Pluggable database altered.
```

```
SQL>
```

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

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

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

SQL>
```

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

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

Pluggable database dropped.

SQL>
```

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

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

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

SQL>
```

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

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

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

```
SQL> set serveroutput on
SQL> DECLARE
 2   compatible BOOLEAN := FALSE;
```

```

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

```

PDB3 is compatible

PL/SQL procedure successfully completed.

SQL>

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

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

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

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

```

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

```

Pluggable database created.

SQL>

- 3. Examine the plugged-in PDB.

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

```

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

```

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

SQL>

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

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

PDB_NAME	STATUS
HRPDB	NEW

SQL>

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

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

OPEN_MODE
MOUNTED

SQL>

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

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

```
SQL>
```

4. Open and connect to HRPDB.

- a. Open HRPDB.

```
SQL> ALTER PLUGGABLE DATABASE HRPDB open;
```

```
Pluggable database altered.
```

```
SQL>
```

- b. Query V\$SERVICES.

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

```
NAME
```

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

```
7 rows selected.
```

```
SQL>
```

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

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

5. Verify the current container name is HRPDB.

```
SQL> SHOW con_name
```

```
CON_NAME
```

```
-----  
HRPDB
```

```
SQL>
```

6. Exit from SQL*Plus.

```
SQL> exit
```

```
...
```

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

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>
```

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

```
SQL> SHOW PDBS

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

3. Close HRPDB.

```
SQL> ALTER PLUGGABLE DATABASE HRPDB CLOSE;

Pluggable database altered.

SQL>
```

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

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

Pluggable database dropped.

SQL>
```

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

```
SQL> SHOW PDBS

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

SQL>
```

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

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


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 ~]$
```

2. 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 ~]$
```

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

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

4. 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  
  2  ADMIN USER pdbtestadmin IDENTIFIED BY DBAdmin_1  
  3  ROLES=(dba);  
Pluggable database created.  
  
SQL>
```

5. Open PDBTEST by using the `ALTER PLUGGABLE DATABASE` command.

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

6. Verify that the open mode for PDBTEST is now READ WRITE. Also, note the container ID for PDBTEST.

```
SQL> col name format a15
SQL> SELECT con_id, name, open_mode FROM v$pdbs;

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

SQL>
```

7. Set the container to PDBTEST.

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

Session altered.

SQL>
```

8. 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.

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

  STATUS          WALLET_TYPE
----- -----
OPEN_NO_MASTER_KEY        AUTOLOGIN

SQL>
```

9. 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
  2 IDENTIFIED BY keystore-password
  3 WITH BACKUP USING 'pdbtest_bkup';

keystore altered.

SQL>
```

10. Query V\$ENCRYPTION_WALLET again to verify that the value in the STATUS column OPEN.

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

          STATUS            WALLET_TYPE
----- -----
OPEN                AUTOLOGIN

SQL>
```

11. You will not need this PDB for any other practices, so you can now drop it.

- a. Set the container to the root.

```
SQL> ALTER SESSION set container = cdb$root;

Session altered.
```

- b. Close PDBTEST.

```
SQL> ALTER PLUGGABLE DATABASE PDBTEST CLOSE;

Pluggable database altered.

SQL>
```

- c. Drop PDBTEST.

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

Pluggable database dropped.

SQL>
```

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

```
SQL> SHOW PDBS

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

SQL>
```

13. 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>
```

2. Verify that the open mode for PDB2 is READWRITE.

```
SQL> col name format a15  
SQL> SELECT con_id, name, open_mode FROM v$pdbs;  
  
CON_ID NAME          OPEN_MODE  
-----  
      2 PDB$SEED      READ ONLY  
      3 PDB1          READ WRITE  
      4 PDB2          READ WRITE  
  
SQL>
```

3. Set the container to PDB2.

```
SQL> ALTER SESSION SET CONTAINER = pdb2;  
  
Session altered.  
  
SQL>
```

4. 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.

```
SQL> SELECT status, wallet_type FROM v$encryption_wallet;  
  
STATUS          WALLET_TYPE  
-----  
OPEN_NO_MASTER_KEY    AUTOLOGIN  
  
SQL>
```

5. 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  
2 IDENTIFIED BY keystore-password  
3 WITH BACKUP USING 'pdb2_bkup';  
  
keystore altered.  
  
SQL>
```

6. Query V\$ENCRYPTION_WALLET again to verify that the STATUS column is set to OPEN.

```
SQL> SELECT status, wallet_type FROM v$encryption_wallet;  
  
STATUS          WALLET_TYPE  
-----  
OPEN            AUTOLOGIN  
  
SQL>
```

7. 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 ~]$
```

2. Source the `oraenv` script.

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

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

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

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

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

CHUNK_TABLESPACE	VARCHAR2(1)
SQL>	

5. List the tablespaces in PDB1.

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

TABLESPACE_NAME
-----
SYSAUX
SYSTEM
TEMP
UNDOTBS1
USERS

SQL>
```

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

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

TABLESPACE_NAME
-----
USERS

SQL>
```

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

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

STATUS      CONTENTS          LOGGING     PLU  BIG  EXTENT_MAN  ALLOCA
-----  -----  -----  -----  -----  -----  -----
ONLINE      PERMANENT        LOGGING    NO   NO   LOCAL       SYSTEM

SQL>
```

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

- `LOGGING` shows the value `LOGGING`, indicating that certain DML operations are logged in the redo log file.
 - `PLUGGED_IN` shows the value `NO`, indicating that the tablespace is not plugged in.
 - `BIGFILE` shows the value `NO`, indicating that the tablespace is a smallfile tablespace.
 - `EXTENT_MANAGEMENT` shows the value `LOCAL`, indicating that the tablespace is locally managed (not dictionary managed).
 - `ALLOCATION_TYPE` shows the value `SYSTEM`, indicating that the extents of the tablespace are managed by the system, and you cannot specify an extent size.
8. List the columns in the `V$TABLESPACE` view by using the `DESCRIBE` command. This view displays tablespace information from the control file.

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

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

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

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

SQL>
```

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

- CON_ID indicates the container to which the data pertains. In this case, PDB1 is container ID 3.
10. List all the tables in the USERS tablespace owned by the HR account.

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

```
TABLE_NAME
```

```
-----  
REGIONS  
LOCATIONS  
DEPARTMENTS  
JOBS  
EMPLOYEES  
JOB_HISTORY
```

```
6 rows selected.
```

```
SQL>
```

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

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

```
INDEX_NAME
```

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

```

REG_ID_PK

19 rows selected.

SQL>

```

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

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

```

SQL>

```

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

```

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

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

SQL>

```

The results show the following:

- AUTOEXTENSIBLE contains the value YES, indicating that the auto extend feature is enabled for a data file. The tablespace size can increase without you having to take any action.
 - BYTES is the size of the file in bytes.
 - MAXBYTES is the maximum file size allowed.
 - USER_BYTES is the size of the file available for user data.
14. Find out how many segments are there in the SYSAUX tablespace by querying the DBA_SEGMENTS view.

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

COUNT(SEGMENT_NAME)
-----
2356

SQL>
```

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

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

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

SQL>
```

16. Exit SQL*Plus.

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

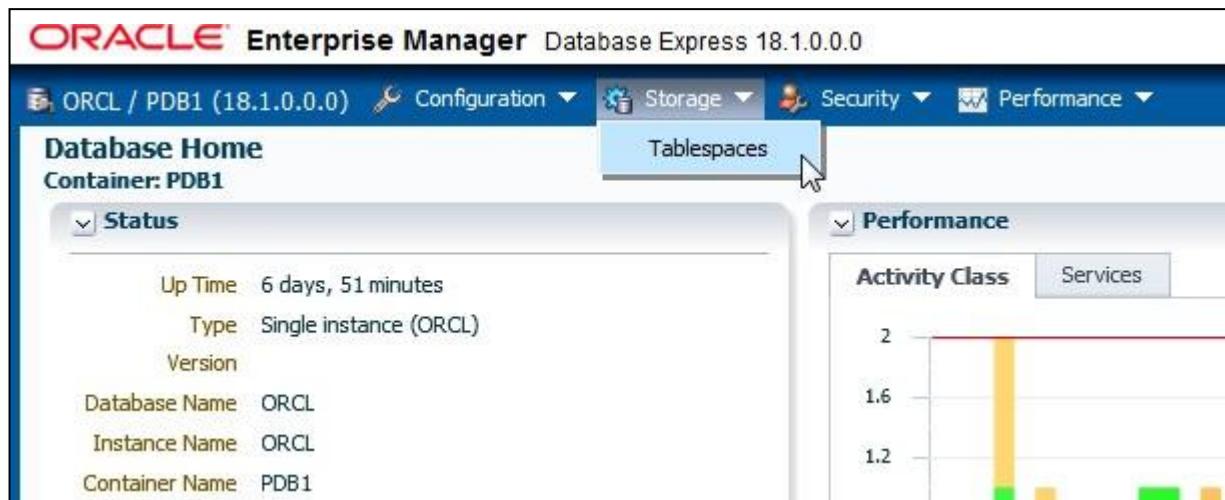
Viewing Tablespace Information by Using Enterprise Manager Database Express

1. Create an SSH tunnel to use the EM Express port (5500) on the compute node of your database deployment.

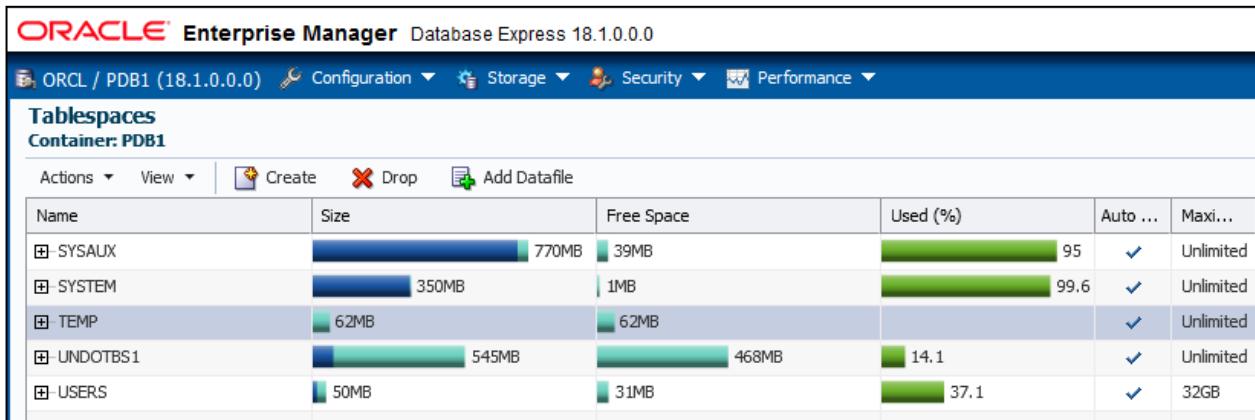
- Open a new terminal window.
- Use `ssh` to create an SSH tunnel.

```
[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 ~]$
```

- Open Firefox. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
- On the Enterprise Manager Database Express login page, enter the following:
 - In the User Name field, enter `PDBADMIN`.
 - In the Password field, enter the password you specified when you created the database deployment.
 - In the Container Name field, enter `PDB1`.
- Click Login.
- Expand **Storage** and then select **Tablespaces**.



- 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.



- 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.
- 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 CreateINVENTORystablespace.sql script to create the INVENTORY tablespace. Next, execute a script named CreateTableX.sql to create and populate a table called x in the INVENTORY tablespace. At first, you will get an error trying to populate the table. In the next section, you correct the problem.

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

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

2. Execute the CreateINVENTORystablespace.sql script.

```
SQL> set echo on  
SQL> @/home/oracle/labs/CreateINVENTORystablespace.sql  
  
SQL> CREATE SMALLFILE TABLESPACE INVENTORY  
      2       DATAFILE  
      3  
      4       '/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF' SIZE 5242880  
      5       DEFAULT NOCOMPRESS  
      6       ONLINE  
      7       SEGMENT SPACE MANAGEMENT AUTO  
      8       EXTENT MANAGEMENT LOCAL AUTOALLOCATE;  
  
Tablespace created.  
  
SQL>
```

3. 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 ~] $
```

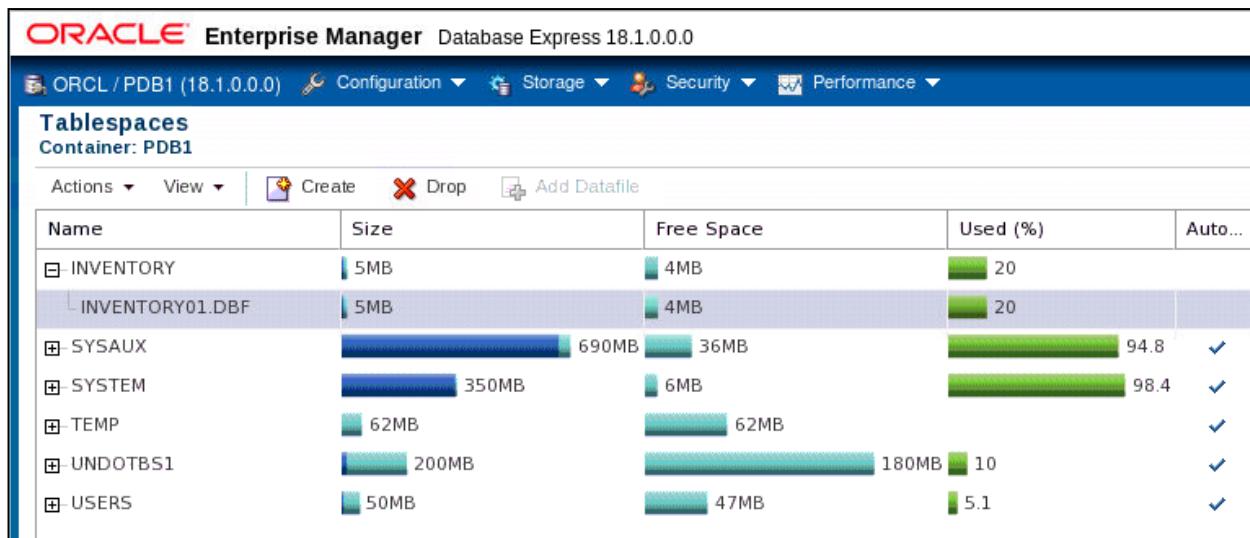
Use EM Database Express to Increase the Size of the INVENTORY01.DBF Data File

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.
 - a. Open a terminal window.
 - b. 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 ~]$
```

2. On your Linux desktop, open Firefox. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
3. On the Enterprise Manager Database Express login page, enter the following:
 - a. In the User Name field, enter PDBADMIN.
 - b. In the Password field, enter the password you specified when you created the database deployment.
 - c. In the Container Name field, enter PDB1.
4. Click **Login**.
5. Expand **Storage** and then select **Tablespaces**.
6. Expand the **INVENTORY** tablespace and select the **INVENTORY01.DBF** data file.



7. Expand Actions and then select **Resize**.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE Enterprise Manager Database Express 18.1.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation is a toolbar with icons for Create, Drop, and Add Datafile. A dropdown menu "Actions" is open, showing options like Create, Drop, Set Status, Tablespace Groups, Add Datafile, Edit Auto Extend, and Resize. The "Resize" option is highlighted with a yellow box. To its right is a table titled "Tablespaces Container: PDB1" with columns for Size and Free Space. The table lists several datafiles with their current sizes and free space.

	Size	Free Space
5MB	4MB	
5MB	4MB	
770MB	39MB	
350MB	1MB	
62MB	62MB	
545MB	468MB	
50MB	31MB	

8. The "Resize Datafile INVENTORY01.DBF" window is displayed. In the Size box, enter **40M**. Don't click OK just yet.



9. Click the **Show SQL** button to view the SQL command that performs the resize action.

- In the Confirmation dialog box, click **OK**.



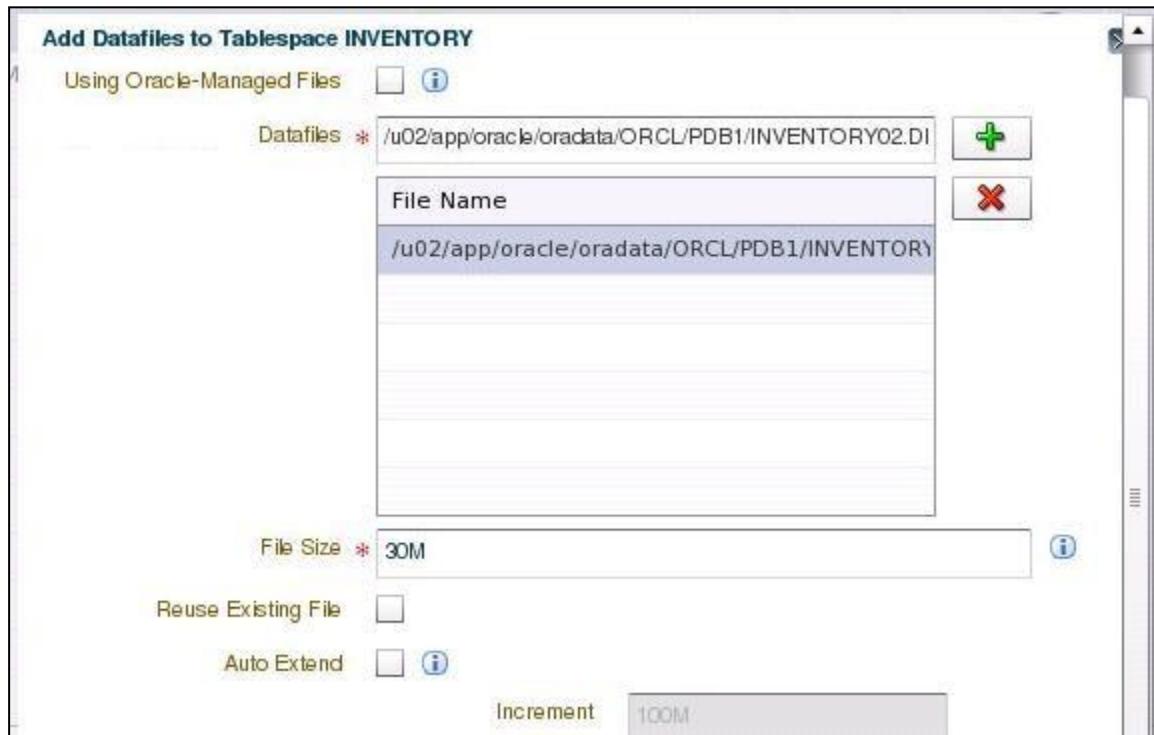
- In the "Resize Datafile INVENTORY01.DBF" dialog box, click **OK**. The SQL code is executed, and the data file is successfully resized.
- In the Confirmation dialog box, click **OK**.
- Verify that the change is reflected in the EM Express interface. The size for the **INVENTORY** tablespace should now be set to 40MB.



Use EM Database Express to Add a Data File to the **INVENTORY** Tablespace

- Select the **INVENTORY** tablespace.
- Expand **Actions** and then select **Add Datafile**.
- The "Add Datafiles to Tablespace INVENTORY" dialog box is displayed.
 - Deselect **Using Oracle-ManagedFiles**.
 - 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.
 - In the File Size field, enter **30M**.

- d. Deselect Auto Extend.



4. Click **Show SQL** and view the SQL code being generated.
5. A Confirmation dialog box displays the code. In the Confirmation dialog box, click **OK**.
6. In the "Add Datafiles to Tablespace INVENTORY" window, click **OK**. A message states that the data file was successfully added to the **INVENTORY** tablespace.
7. In the Confirmation dialog box, click **OK**.
8. Expand the **INVENTORY** tablespace and verify that it now has two data files: **INVENTORY01.DBF** and **INVENTORY02.DBF**.



9. 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>
```

3. 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 ~] $
```

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

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

5. 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>
```

2. 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>
```

2. Use the `SHOW PARAMETER` command to verify that tablespace encryption by default is configured.

```
SQL> SHOW PARAMETER encrypt  
  
NAME                      TYPE        VALUE  
-----  
-----  
encrypt_new tablespaces    string      CLOUD_ONLY  
SQL>
```

3. Set the container to `PDB1`.

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

4. Check whether any of the tablespaces in `PDB1` are encrypted.

```
SQL> SELECT tablespace_name, encrypted FROM dba_tablespaces;  
  
TABLESPACE_NAME          ENC  
-----  
-----  
SYSTEM                   NO  
SYSAUX                  NO  
UNDOTBS1                NO  
TEMP                     NO  
USERS                   YES  
  
SQL>
```

5. 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.

6. 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>
```

7. Question: Is the new tablespace encrypted?

```
SQL> SELECT tablespace_name, encrypted FROM dba_tablespaces;  
  
TABLESPACE_NAME          ENC  
-----  
SYSTEM                  NO  
SYSAUX                 NO  
UNDOTBS1                NO  
TEMP                   NO  
USERS                  YES  
TESTENCRYPT              YES  
  
6 rows selected.  
  
SQL>
```

Answer: Yes.

8. Drop the TESTENCRYPT tablespace.

```
SQL> DROP TABLESPACE testencrypt INCLUDING CONTENTS AND  
DATAFILES;  
  
Tablespace dropped.  
  
SQL>
```

9. 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 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 ~]$
```

2. Source the `oraenv` script.

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

3. Connect to `PDB1` as the `SYSTEM` user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
$ sqlplus SYSTEM/password@PDB1  
...  
SQL>
```

4. 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>
```

5. Check the database-wide threshold values for the Tablespace Space Usage metric.

- a. Connect to the root container.

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

- b. 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.

```
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;  
  
WARNING_VALUE      CRITICAL_VALUE  
-----  
85                  97  
SQL>
```

6. 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.

- a. Connect to PDB1.

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

- b. 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  
2  DATAFILE '/u02/app/oracle/oradata/ORCL/PDB1/tbsalert.dbf'  
3  SIZE 120M REUSE LOGGING EXTENT MANAGEMENT LOCAL
```

```
4 SEGMENT SPACE MANAGEMENT AUTO;
```

Tablespace created.

```
SQL>
```

7. Query how much free space the TBSALERT tablespace holds by executing the \$HOME/labs/TBSALERT_free_space.sql script.

```
SQL> set echo on
SQL> @/home/oracle/labs/TBSALERT_free_space.sql
SQL> SELECT df.tablespace_name tablespace, fs.bytes free,
2 df.bytes , fs.bytes *100/ df.bytes PCT_FREE
3 FROM dba_data_files df ,dba_free_space fs
4 WHERE df.tablespace_name = fs.tablespace_name
5 AND df.tablespace_name = 'TBSALERT';

TABLESPACE          FREE      BYTES      PCT_FREE
-----  -----  -----
TBSALERT           124780544  125829120  99.1666667

SQL>
```

8. 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>
```

9. 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.

```
SQL> SELECT warning_value, critical_value FROM dba_thresholds
WHERE object_name='TBSALERT';

WARNING_VALUE      CRITICAL_VALUE
-----  -----
55                  70
```

```
SQL>
```

10. Query the REASON and RESOLUTION columns from the DBA_ALERT_HISTORY view for the TBSALERT tablespace.

```
SQL> col reason format a60
SQL> SELECT reason, resolution FROM dba_alert_history WHERE
object_name='TBSALERT';

REASON                                     RESOLUT
-----
Threshold is updated on metrics "Tablespace Space Usage" cleared

SQL>
```

11. Exit SQL*Plus.

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

12. 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.
```

```
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 6 7 8 9 10 11  
PL/SQL procedure successfully completed.  
  
SQL>  
109568 rows created.  
  
SQL>  
109568 rows created.  
  
SQL>  
109568 rows created.  
  
SQL>  
Commit complete.  
  
SQL> Disconnected ...  
[oracle@MYDBCS ~]$
```

13. Check the fullness level of the TBSALERT tablespace to see if the warning level has been reached.

- a. Start SQL*Plus and connect to PDB1 as the SYSTEM user.

```
$ sqlplus SYSTEM/password@PDB1  
...  
SQL>
```

- b. 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>

```

- c. 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.

```

SQL> set echo on
SQL> @$HOME/labs/TBSALERT_free_space
SQL> SELECT df.tablespace_name tablespace, fs.bytes free,
2 df.bytes, fs.bytes *100/ df.bytes PCT_FREE
3 FROM dba_data_files df, dba_free_space fs
4 WHERE df.tablespace_name = fs.tablespace_name
5 AND df.tablespace_name = 'TBSALERT';

TABLESPACE          FREE        BYTES      PCT_FREE
-----  -----
TBSALERT           49283072   125829120  39.1666667

SQL>

```

- d. Wait a few minutes.
- e. 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>
```

2. Wait a few minutes.
3. Query the fullness of the tablespace. The result shows that the tablespace is 75% full.

```
SQL> SELECT sum(bytes) * 100 / 125829120  
  2  FROM    dba_extents  
  3  WHERE   tablespace_name='TBSALERT';  
  
SUM(BYTES)*100/125829120  
-----  
75  
  
SQL>
```

4. 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>

5. 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>

6. 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**
2 FROM dba_extents
3 WHERE tablespace_name='TBSALERT';

SUM(BYTES)*100/125829120

75

```
SQL>
```

Create a Segment Advisor Task

7. Create a Segment Advisor task to get recommendations about the current space situation by executing the `$HOME/labs/seg_advsr_task.sql` script.

```
SQL> set echo on
SQL> @$HOME/labs/seg_advsr_task.sql
SQL> DECLARE
  2   tname VARCHAR2(128) := 'my_seg_task';
  3   tname_desc  VARCHAR2(128) := 'Get shrink advice for
segments in TBSALERT';
  4   task_id NUMBER;
  5   object_id NUMBER;
  6   objectname  VARCHAR2(100);
  7   objecttype  VARCHAR2(100);
  8   BEGIN
  9   dbms_advisor.create_task('Segment Advisor',
task_id,tname,tname_desc,NULL);
 10  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>
```

8. Execute the task.

```
SQL> DECLARE
  2   tname VARCHAR2(128) := 'my_seg_task';
  3   BEGIN
  4   dbms_advisor.EXECUTE_TASK(tname);
  5   END;
  6 /
```

PL/SQL procedure successfully completed.

```
SQL>
```

9. 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>

```

10. 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
   2  FROM dba_advisor_findings f, dba_advisor_objects o
   3  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>

```

11. 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>
```

12. 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
  2  FROM    dba_extents
  3 WHERE   tablespace_name='TBSALERT';

SUM(BYTES)*100/125829120
-----
30.15625

SQL>
```

13. Drop the TBSALERT tablespace.

```
SQL> DROP TABLESPACE tbsalert INCLUDING CONTENTS AND DATAFILES;

Tablespace dropped.

SQL>
```

14. 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 ~]$
```

2. 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>
```

3. 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.

```
SQL> col index_name format a20  
SQL> SELECT index_name, compression FROM user_indexes WHERE  
index_name = 'I_TEST';  
  
INDEX_NAME          COMPRESSION  
-----  
I_TEST              DISABLED  
  
SQL>
```

4. 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>
```

5. Exit SQL*Plus, but keep the terminal window open.

```
SQL> EXIT
```

6. 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.

```
$ cat $ORACLE_HOME/rdbms/admin/dbmscomp.sql  
...  
Rem  
Rem      NAME  
Rem      dbmscomp.sql - DBMS Compression package  
Rem  
Rem      DESCRIPTION  
Rem      Contains package specification for the wrapper  
dbms_compression
```

```

Rem      package and internal prvt_compression package. We
integrate these
Rem      packages with the advisor framework.
Rem
...
grant execute on dbms_compression to public
/
show errors;

@?/rdbms/admin/sqlsessend.sql
[oracle@MYDBCS ~]$
```

7. Start SQL*Plus and connect to PDB1 as the HR user.

```

$ sqlplus hr/password@PDB1
...
SQL>
```

8. 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_ncmp pls_integer;
row_cmp pls_integer;
row_ncmp 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_ncmp => blkcnt_ncmp,
row_cmp => row_cmp,
```

```

row_ncmp => row_ncmp,
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_ncmp);
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>

9. 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_ncmp pls_integer;
row_cmp pls_integer;
row_ncmp 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_ncmp => blkcnt_ncmp,
row_cmp => row_cmp,
row_ncmp => row_ncmp,
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_ncmp);
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>
```

10. 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.
11. Rebuild the index with Advanced High Compression.

```

SQL> ALTER INDEX hr.i_test REBUILD COMPRESS ADVANCED HIGH;

Index altered.

SQL>
```

12. 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
```

```

SQL> SELECT index_name, compression FROM user_indexes WHERE
index_name = 'I_TEST';

INDEX_NAME          COMPRESSION
-----
I_TEST              ADVANCED HIGH

SQL>

```

13. 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>

```

14. Question: Is it possible to revert back to the initial compression level?

Answer: Yes.

15. Revert back to the initial compression level.

```

SQL> ALTER INDEX hr.i_test REBUILD NOCOMPRESS;

Index altered.

SQL>

```

16. 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>

```

17. 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>
```

2. 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>
```

3. 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>
```

4. Execute the \$HOME/labs/CreateINVENTORYTablespace.sql script to create an unpopulated tablespace named INVENTORY.

```
SQL> @$HOME/labs/CreateINVENTORYTablespace.sql

Tablespace created.

SQL>
```

5. 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 ~]$
```

6. 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.
 - a. Question: Are the rows that were inserted into the table lost or definitely inserted?
Answer: 2048 rows were inserted.
 - b. 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.
7. Start SQL*Plus again and connect to PDB1 as the PDBADMIN user.

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

8. Enable resumable mode.

```
SQL> ALTER SESSION ENABLE RESUMABLE;  
  
Session altered.  
  
SQL>
```

9. Re-execute the CreateTable_Xscript. 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 ;

```

10. 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.

11. 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 ~]$

```

2. Source the oraenv script.

```

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

```

3. Start SQL*Plus and connect to PDB1 as SYSTEM.

```

$ sqlplus SYSTEM/password@PDB1
...
SQL>

```

4. 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>

```

5. Exit SQL*Plus, but keep the terminal window open.

```

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

```

6. 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 ~] $
```

7. Proceed with the appropriate corrective action. Because the INVENTORY tablespace is not autoextensible, you can configure it as autoextensible with a size limit.
- Start SQL*Plus and connect to PDB1 as the SYSTEM user.

```
$ sqlplus SYSTEM/password@PDB1  
...  
SQL>
```

- 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>
```

- 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>
```

- 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 ~] $
```

2. 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>
```

2. 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 ~]$
```

2. Source the `oraenv` script.

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

3. 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 ~]$
```

4. 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>
```

4. 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.

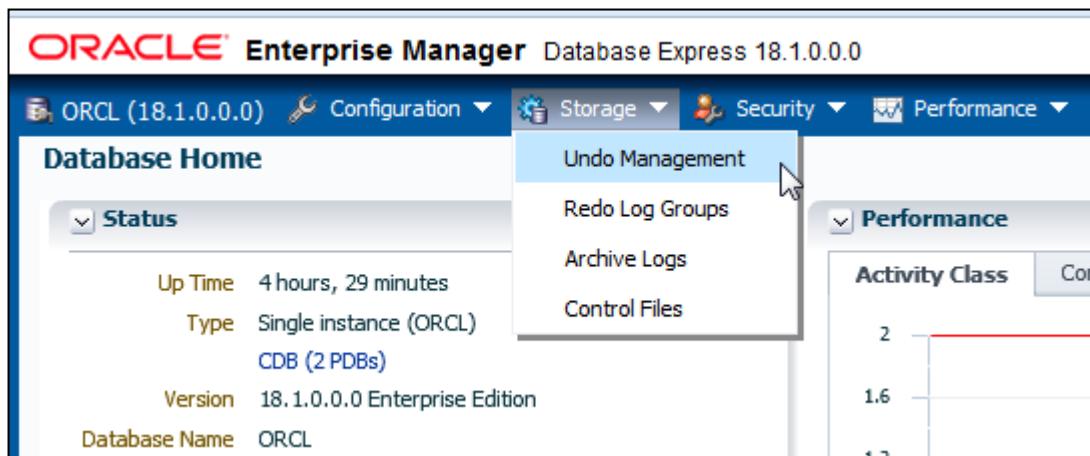
```
[oracle@MYDBCS ~] $ $HOME/labs/UNDO_loop.sh
```

Use Enterprise Manager Database Express to Analyze Undo Data

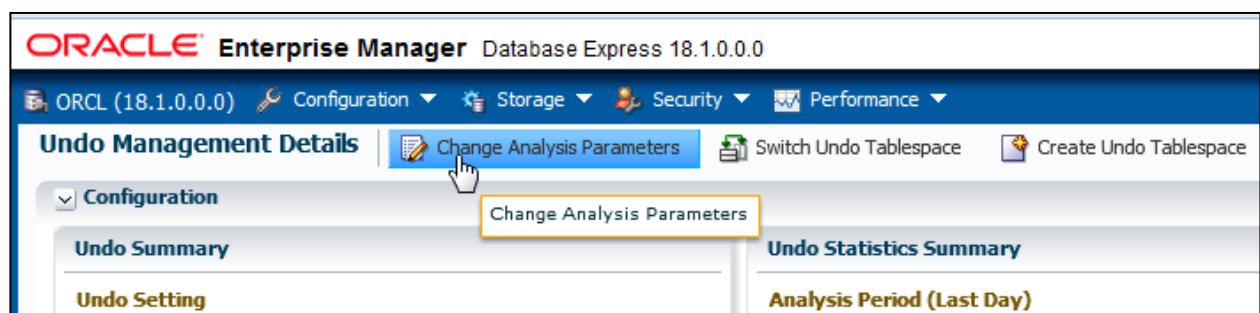
1. Open a new terminal window and create an SSH tunnel.

```
[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 ~] $
```

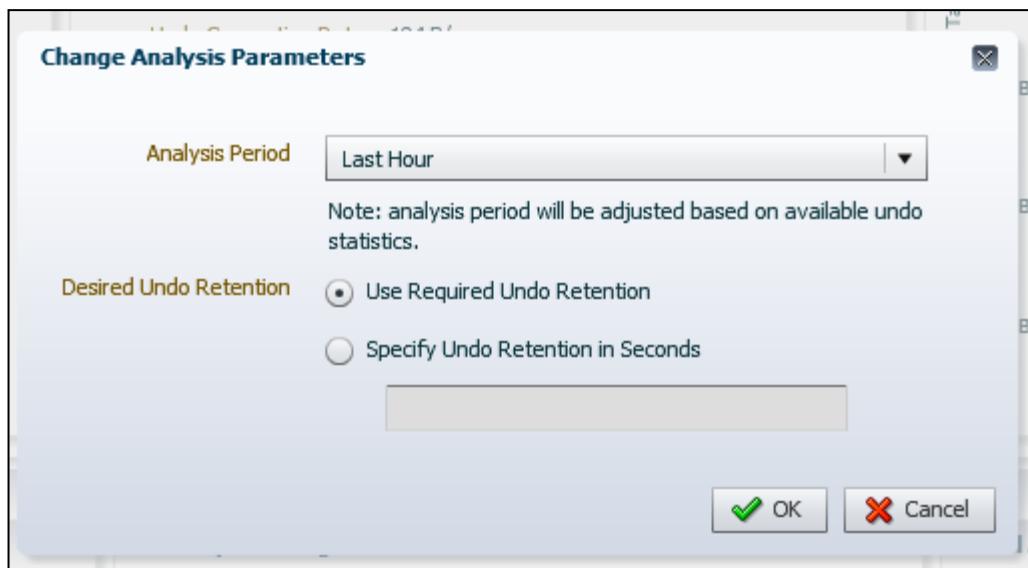
2. Open a browser and launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
3. Log in as the **SYSTEM** user. Refer to *Course Practice Environment: Security Credentials* for the password value.
4. Select **Storage** and then **Undo Management**.



5. Click **Change Analysis Parameters**.



6. In the Analysis Period drop-down list, select **Last Hour**. Leave **Use Required Undo Retention** selected and click **OK**.

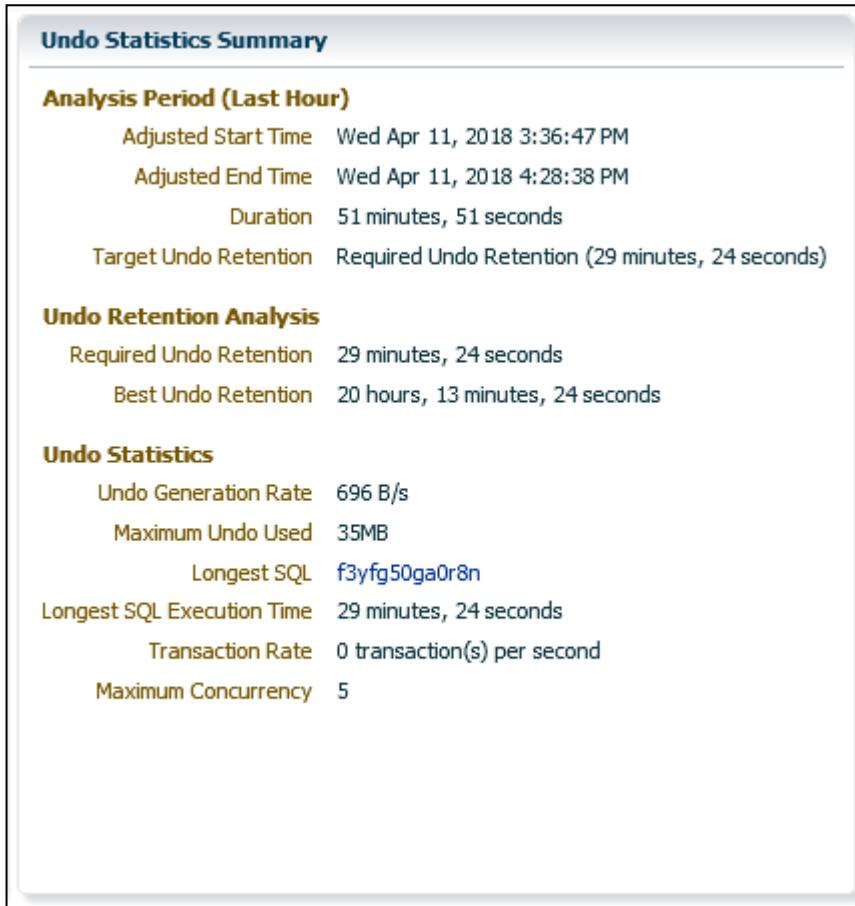


7. In the Confirmation dialog box, click **OK**.
8. In the Undo Summary pane, review the undo configuration. Your tablespace size may be different.

The pane shows the following sections:

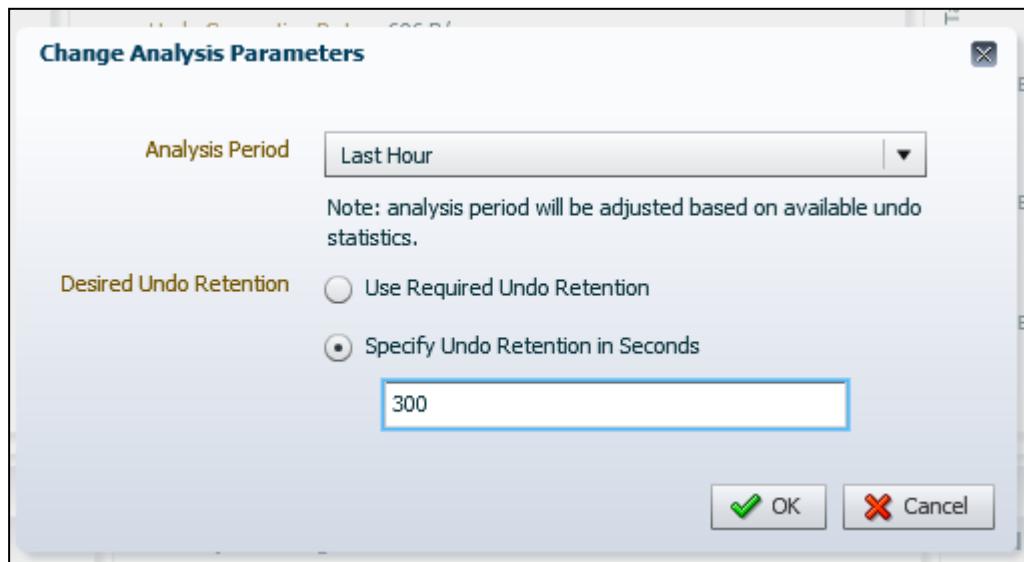
- Undo Setting**: Undo Management is set to auto, and the Low Undo Retention Threshold is 900s.
- Tablespace**: The tablespace is UNDOTBS1, with Retention Guaranteed set to Yes, Size at 60MB (41.6% free), and Auto Extensible set to No.
- Errors and Warnings**: Snapshot Too Old Errors, Out of Space Errors, and Unexpired Blocks Stolen are all 0.
- Advisor Findings**: Both Health and Setting show 'No problems'.

9. 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.
10. In the Undo Statistics Summary pane, review the information. Your values will be different from those shown below.



11. 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.
12. 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.
13. Restart an analysis, keeping the past hour as the period of analysis and defining the Undo Retention to 300 seconds (5 minutes).
 - a. Click **Change Analysis Parameters**.

- b. 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**.



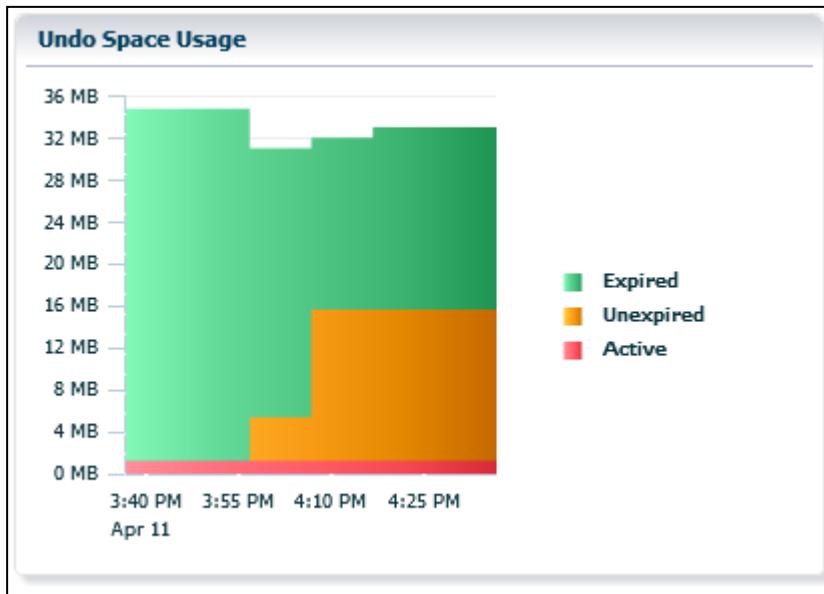
- c. In the Confirmation dialog box, click **OK**.
- d. In the Undo Statistics Summary pane, review the information (particularly the Undo Retention Analysis section).

Undo Statistics Summary	
Analysis Period (Last Hour)	
Adjusted Start Time	Wed Apr 11, 2018 3:36:47 PM
Adjusted End Time	Wed Apr 11, 2018 4:31:33 PM
Duration	54 minutes, 46 seconds
Target Undo Retention	Required Undo Retention (29 minutes, 24 seconds)
Undo Retention Analysis	
Required Undo Retention	29 minutes, 24 seconds
Best Undo Retention	21 hours, 7 minutes, 18 seconds
Undo Statistics	
Undo Generation Rate	666 B/s
Maximum Undo Used	35MB
Longest SQL	f3yfg50ga0r8n
Longest SQL Execution Time	29 minutes, 24 seconds
Transaction Rate	0 transaction(s) per second
Maximum Concurrency	5

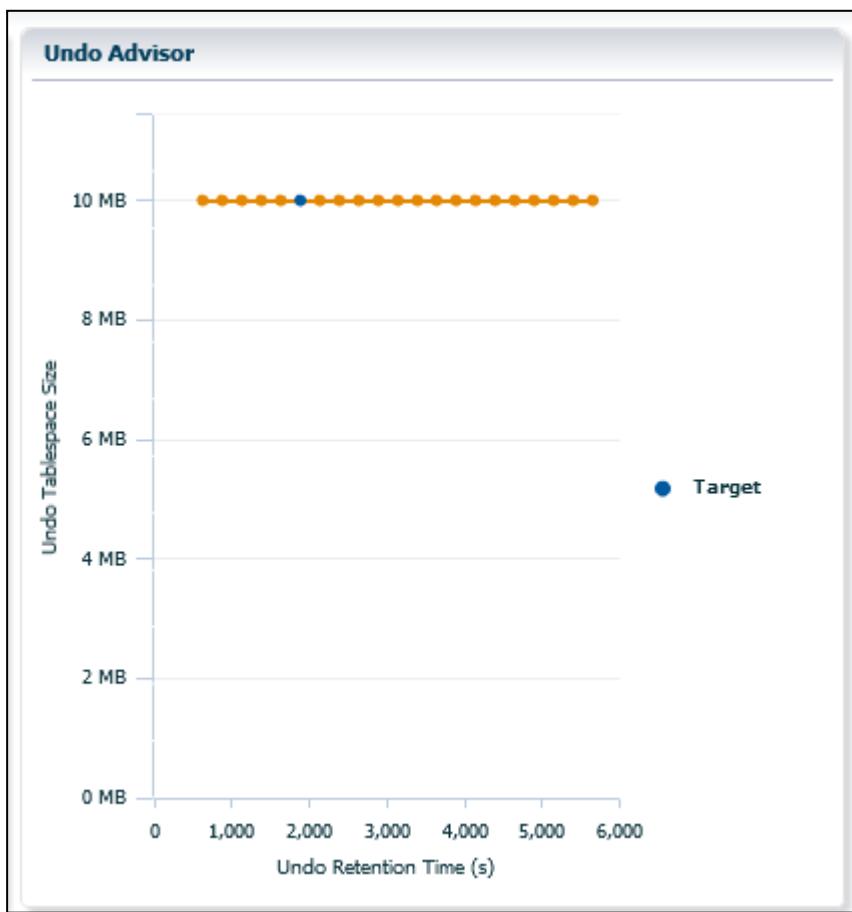
- e. Is the best undo retention time the same as before?

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.

- f. 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.



- g. 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.



- h. 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;
```

2. 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 ~]$
```

2. Source the `oraenv` script.

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

3. Create a new directory named PDB2.

```
[oracle@MYDBCS ~]$ mkdir /u02/app/oracle/oradata/ORCL/PDB2  
[oracle@MYDBCS ~]$
```

4. 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 ~]$
```

5. 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 ~]$
```

6. 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.

7. 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>
```

8. 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>
```

9. 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>
```

10. Exit SQL*Plus.

```
SQL> EXIT
```

```
...
```

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

11. 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
```

```

. . exported "OE"."ORDERS"                      12.73
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. Use the `oerr` utility to determine whether the ORA-39173 error is of concern.

```

[oracle@MYDBCS ~]$ oerr ora 39173
39173, 00000, "Encrypted data has been stored unencrypted in
dump file set."
// *Cause: No encryption password was specified for an export
job that
//           involved data that was encrypted in the database.
// *Action: No specific user action is required. This is only a
warning that
//           secure data may be readable from within the dump
file set.
[oracle@MYDBCS ~]$

```

13. 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.

14. 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 ~]$

```

15. 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 INITTRANS 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>
```

2. 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>
```

3. Exit SQL*Plus.

```
SQL> EXIT
...
```

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

4. 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 ~]$
```

5. 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.

6. Connect to PDB2 as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
$ sqlplus SYSTEM/password@PDB2  
...  
SQL>
```

7. Create the DP_FOR_OE directory in PDB2.

```
SQL> CREATE DIRECTORY dp_for_oe AS  
'/u01/app/oracle/admin/ORCL/dpdump' ;
```

```
Directory created.
```

```
SQL>
```

8. Exit SQL*Plus.

```
SQL> EXIT  
...
```

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

9. 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 INITTRANS 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 ~]$
```

10. Question: Did the import complete successfully?

Answer: Not completely. Data Pump imported only the objects that it could process without any error.

11. 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.

12. Create the TBS_APP2 tablespace in PDB2.

- 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>
```

- 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>
```

- Exit SQL*Plus.

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

13. 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 ~] $
```

14. 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 OTEST Schema in PDB2

Verify that the new OTEST schema exists in PDB2.

1. Start SQL*Plus and connect to PDB2 as the OTEST user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
$ sqlplus oetest/password@PDB2
...
SQL>
```

2. View the list of tables to which the OTEST user has access.

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
-----
ORDERS
ORDER_ITEMS

SQL>
```

3. 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>
```

4. List the indexes to which the OTEST user has access.

```
SQL> SELECT index_name FROM user_indexes;

INDEX_NAME
-----
SYS_C0011373
I_ORDER_ITEMS
```

```
SQL>
```

5. List the sequences to which the OETEST user has access.

```
SQL> SELECT sequence_name FROM user_sequences;
```

```
SEQUENCE_NAME
```

```
-----
```

```
ORDERS_SEQ
```

```
SQL>
```

6. Exit SQL*Plus.

```
SQL> EXIT
```

```
...
```

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

7. 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>
```

2. 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>
```

3. Drop the target user created in the previous import operation.

```
SQL> DROP USER oetest CASCADE;
```

```
User dropped.
```

```
SQL>
```

4. Exit SQL*Plus.

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

5. 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 ~]$
```

6. Verify that the OE schema was imported as OETEST into PDB2.
 - a. 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>
```

- b. 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>
```

- c. Query the number of rows in the ORDER_ITEMS table. The table has 665 rows.

```
SQL> SELECT count(*) FROM order_items;

COUNT(*)
-----
665

SQL>
```

- d. Exit SQL*Plus.

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

7. 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
```

2. 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 ~]$
```

3. 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>
```

4. 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>
```

5. Exit SQL*Plus.

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

6. Change to the /home/oracle/labs directory.

```
[oracle@MYDBCS ~]$ cd /home/oracle/labs
```

7. 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]$
```

8. 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>
```

9. Verify that the table is loaded with the eight records from the products.dat file. The results show that the records were loaded.

```
SQL> SELECT * FROM products;  
  
PRODUCT_ID COU_LABEL DETAILED_LABEL  
----- -----  
1001 ENG Shutter1 Outdoor shutter1  
1002 FRE Porte1 Porte exterieure1  
1003 SPA Puertal 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  
  
14 rows selected.  
  
SQL>
```

10. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@MYDBCS labs]$
```

11. 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 Datatype	Position	Len	Term	Encl
PRODUCT_ID CHARACTER	FIRST	*	,	
COUNTRY CHARACTER	NEXT	*	,	
LABEL CHARACTER	NEXT	*	,	
DETAILED_LABEL CHARACTER	NEXT	*	,	

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]$

```

12. 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.

13. 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]$
```

2. 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>
```

3. Determine the number of rows in the SH.INVENTORIES table. The result shows 476 rows.

```
SQL> SELECT count(*) FROM inventories;

COUNT(*)
-----
476

SQL>
```

4. Exit from SQL*Plus.

```
SQL> EXIT
...
```

```
[oracle@MYDBCS labs]$
```

5. 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.

```
[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.
```

```
Path used:      Conventional
```

```
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]$
```

6. 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>
```

7. Determine the number of rows in the SH.INVENTORIES table. The result shows 559 rows.

```
SQL> SELECT count(*) FROM inventories;
```

```
COUNT (*)
```

```
-----
```

```
559
```

```
SQL>
```

8. 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.

9. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@MYDBCS labs]$
```

10. View the inventories.log file. Notice that the insert option in effect for the SH.INVENTORIES table is APPEND.

```
$ 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  
  
Column Name Position Len Term Encl  
Datatype  
-----  
-----  
WAREHOUSE_ID FIRST * ,  
CHARACTER
```

```

PRODUCT_ID          NEXT    *   ,
CHARACTER
QUANTITY_ON_HAND   NEXT    *   ,
CHARACTER

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]$

```

11. 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)
$
```

12. 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)

$
```

13. 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.

```
[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
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

Table SH.INVENTORIES:
 83 Rows successfully loaded.

Check the log file:
```

```
inventories.log  
for more information about the load.  
[oracle@MYDBCS labs]$
```

14. 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>
```

15. 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]$
```

2. 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.

```
[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.

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

Table SH.INVENTORIES:
 20 Rows successfully loaded.

Check the log file:
  inventories.log
for more information about the load.
[oracle@MYDBCS labs]$
```

3. 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
```

(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 Datatype	Position	Len	Term	Encl
WAREHOUSE_ID CHARACTER	FIRST	*	,	
PRODUCT_ID CHARACTER	NEXT	*	,	
QUANTITY_ON_HAND CHARACTER	NEXT	*	,	

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.

```

0 Rows not loaded because all fields were null.

Space allocated for bind array: 7740 bytes (10 rows)
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]$

```

4. 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.

```

[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.

Path used: Direct
Save data point reached - logical record count 10.
Save data point reached - logical record count 20.
Save data point reached - logical record count 30.

```

```
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]$
```

2. 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.

3. 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.

4. Question: Does SQL*Loader in direct mode ignore all constraints?

Answer: No, it does not. It enforces `PRIMARY KEY`, `UNIQUE`, and `NOT NULL` constraints.

5. Disconnect from the compute node and close the terminal window.

Practices for Lesson 16:
Backup and Recovery
Concepts

This document should not be distributed.

Practices for Lesson 16

There are no practices for this lesson.

This document should not be distributed.

Practices for Lesson 17:
Backup and Recovery
Configuration

This document should not be distributed.

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 ~]$
```

2. Source the `oraenv` script.

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

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

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

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

```
SQL> SELECT name FROM v$controlfile;  
  
NAME  
-----  
/u02/app/oracle/oradata/ORCL/control01.ctl  
/u03/app/oracle/fast_recovery_area/ORCL/control02.ctl
```

```
SQL>
```

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

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

```
SQL> SHOW PARAMETER control_files

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

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

```
SQL> CREATE PFILE FROM SPFILE;

File created.

SQL>
```

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

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

8. Exit SQL*Plus.

```
SQL> EXIT
```

9. Create a directory for the new control file.

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

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

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

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

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

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

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

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

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

14. Start the database instance.

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

15. View the `CONTROL_FILES` parameter again.

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

```
SQL>
```

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

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

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

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

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

- b. Exit SQL*Plus.

```
SQL> EXIT
```

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

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

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

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

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

- b. Create the SPFILE.

```
SQL> CREATE SPFILE FROM PFILE;

File created.

SQL>
```

19. Start the database instance.

```
SQL> STARTUP
ORACLE instance started.

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

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

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

```
SQL> SHOW PARAMETER control_files

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

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

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

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

SQL>
```

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/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
/dev/mapper/redoVolGroup-lvol0
                      25G   3.3G   20G  14% /u04
[oracle@MYDBCS ~]$
```

2. If the used storage (Use%) on /u03 is more than 50%, add storage to the database deployment.
 - a. Close the connection to the compute node.
 - b. Open the browser and access the Oracle Cloud web site by entering <https://cloud.oracle.com/home>.
 - c. Click **Sign In**.
 - d. 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**.
 - e. On the Oracle Cloud Account Sign In page, enter the user name and password as provided. Click **Sign In**.
 - f. Expand the menu next to Oracle Cloud My Services.
 - g. Expand the **Services** menu and click **Database**.
 - h. Search for your database deployment (instance) by entering its name in the search field.
 - i. Click the link on the database deployment (instance) name.

- j. Expand the menu in the Resources area and select **Scale Up/Down**.

- k. 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.

- l. Click **Yes, Scale Up/Down**.
m. A confirmation message is displayed. Periodically refresh the page until you see that the status is once again Ready.
n. Sign out of your Cloud account.
3. 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 ~]$
```

4. 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
/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
                      56G   6.3G   47G  12% /u03
[oracle@MYDBCS ~]$
```

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.

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

```
SQL> SHOW PARAMETER db_recovery_file_dest

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

3. Question: Is the fast recovery area enabled?

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

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

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

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

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

6. 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.

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

```
SQL> SHOW PARAMETER db_recovery_file_dest_size
```

NAME	TYPE	VALUE
db_recovery_file_dest_size	big integer	12G

```
SQL>
```

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>
```

2. Question: Why is it recommended to have three groups when two would be sufficient?
Answer: The Oracle Database server treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next. After all groups have been written to, the Oracle Database server begins overwriting information in the first log group. If the database is configured in **ARCHIVELOG** mode, the LGWR cannot overwrite data in the first log group if it has not been archived.
3. Question: Can multiplexing redo logs impact database performance?
Answer: Multiplexing redo logs may heavily influence database performance because a commit cannot complete until the transaction information has been written to the logs by LGWR. You must place your redo log files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files. Because only one group is written to at a given time, there is no performance impact in having members from several groups on the same disk.

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

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

Database altered.

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

Database altered.

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

Database altered.

SQL>
```

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

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

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

 6 rows selected.

SQL>
```

6. Switch the log files and observe the changes.
- Find out which log group is the current log group. In this example, the query result shows that group 3 is the current group. Your current group may be different.

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

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

```
SQL>
```

- 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>
```

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

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

GROUP#	STATUS	MEMBER
1	/u03/app/oracle/fast_recovery_area/ORCL/redo01b.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	

```
6 rows selected.
```

```
SQL>
```

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

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

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

```
SQL>
```

- e. Switch the log file.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

```
System altered.
```

```
SQL>
```

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

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

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

```
SQL>
```

- g. Switch the log file again.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

```
System altered.
```

```
SQL>
```

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

```

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

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

SQL>

```

- i. Question: Can the LGWR background process write to only one member of the CURRENT group in case the other members are missing or damaged?
Answer: Yes, it can. As long as there is one member left in the CURRENT group, LGWR can work.
- 7. To save space in your course practice environment, drop the redo log file members you created in step 4.
 - a. Determine which redo log group is current. You cannot drop a member of the current group.

```

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

      GROUP# STATUS
----- 
          1 INACTIVE
          2 CURRENT
          3 INACTIVE

SQL>

```

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

```

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

Database altered.
SQL> alter system switch logfile;

System altered.

SQL>

```

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

```

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

Database altered.

```

```
SQL> alter system switch logfile;  
  
System altered.  
  
SQL>
```

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

```
SQL> ALTER DATABASE DROP LOGFILE MEMBER  
'/u03/app/oracle/fast_recovery_area/ORCL/redo03b.log';  
  
Database altered.  
SQL> alter system switch logfile;  
  
System altered.  
  
SQL>
```

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

```
SQL> SELECT group#, members, archived, status FROM v$log;  
  
          GROUP#    MEMBERS  ARC  STATUS  
-----  
           1          1  YES  ACTIVE  
           2          1  NO   CURRENT  
           3          1  YES  ACTIVE  
  
SQL>
```

- f. Exit from SQL*Plus.

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

- g. Remove the physical files from the operating system.

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

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

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

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
```

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

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

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

3. Exit from SQL*Plus.

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


**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 ~]$
```

2. Source the `oraenv` script.

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

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

```
[oracle@MYDBCS ~]$ sqlplus system/password  
...  
SQL>
```

4. 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>
```

5. Back up the control file to a trace file.

```
SQL> ALTER DATABASE BACKUP controlfile TO trace;
```

```
Database altered.
```

```
SQL>
```

6. Exit SQL*Plus.

```
SQL> EXIT
```

```
...
```

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

7. 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] $
```

8. 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] $
```

9. 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] $
```

10. 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'

```

```

      SIZE 65011712  REUSE AUTOEXTEND ON NEXT  655360  MAXSIZE
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'
      SIZE 65011712  REUSE AUTOEXTEND ON NEXT  655360  MAXSIZE
32767M;
ALTER SESSION SET CONTAINER = CDB$ROOT;
...
[oracle@MYDBCS trace]$

```

11. Open an editor and paste the code into a new file named `ControlFileBackup.sql` in the `/home/oracle` directory.

12. 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).

13. 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.

14. 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.

15. Question: Which other attributes structure the `ORCL` database?

Answer: The ARCHIVELOG mode, the character set, and the name of the CDB.

16. 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.

17. 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>
```

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

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

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

3. Question: In your configuration, does RMAN automatically back up the control file and server parameter file (SPFILE) with every backup and database structural change?
Answer: Yes, it does because the CONTROLFILE AUTOBACKUP attribute is set to ON.
4. Question: Will a backup operation back up all control files or only one of the multiplexed control files?
Answer: It will back up only one of the multiplexed control files because all control files in a database are identical.
5. Exit RMAN.

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

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/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
/dev/mapper/redoVolGroup-lvol0
                      25G   3.3G   20G  14% /u04
[oracle@MYDBCS ~]$
```

2. If the used storage (Use%) on /u03 is more than 50%, add storage to the database deployment.
 - a. Close the connection to the compute node.
 - b. Open the browser and access the Oracle Cloud web site by entering <https://cloud.oracle.com/home>.
 - c. Click **Sign In**.
 - d. 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**.
 - e. On the Oracle Cloud Account Sign In page, enter the user name and password as provided. Click **Sign In**.
 - f. Expand the menu next to Oracle Cloud My Services.
 - g. Expand the **Services** menu and click **Database**.
 - h. Search for your database deployment (instance) by entering its name in the search field.
 - i. Click the link on the database deployment (instance) name.

- j. Expand the menu in the Resources area and select **Scale Up/Down**.

- k. 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.

- I. Click **Yes, Scale Up/Down**.
- m. A confirmation message is displayed. Periodically refresh the page until you see that the status is once again Ready.
- n. Sign out of your Cloud account.
3. 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 ~]$
```

4. 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
/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
                      56G   6.3G   47G  12% /u03
[oracle@MYDBCS ~]$
```

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>
```

2. View the structure of the CDB in terms of PDBs, tablespaces, and data files (permanent and temporary). Your file numbers will differ from those shown below.

```
RMAN> REPORT schema;  
  
using target database control file instead of recovery catalog  
Report of database schema for database with db_unique_name ORCL  
  
List of Permanent Datafiles  
=====
```

File	Size (MB)	Tablespace	RB	Segs	Datafile Name
1	870	SYSTEM	YES		/u02/app/oracle/oradata/ORCL/system01.dbf

```

3      1480      SYSAUX          NO
/u02/app/oracle/oradata/ORCL/sysaux01.dbf
4       60      UNDOTBS1        YES
/u02/app/oracle/oradata/ORCL/undotbs01.dbf
5      340      PDB$SEED:SYSTEM    NO
/u02/app/oracle/oradata/ORCL/pdbseed/system01.dbf
6      620      PDB$SEED:SYSAUX    NO
/u02/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf
7       5      USERS            NO
/u02/app/oracle/oradata/ORCL/users01.dbf
8      200      PDB$SEED:UNDOTBS1   NO
/u02/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf
12     350      PDB1:SYSTEM        YES
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf
13     790      PDB1:SYSAUX        NO
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf
14     200      PDB1:UNDOTBS1      YES
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf
15     50      PDB1:USERS         NO
/u02/app/oracle/oradata/ORCL/PDB1/PDB1_users01.dbf
19     350      PDB2:SYSTEM        YES
/u02/app/oracle/oradata/ORCL/PDB2/system01.dbf
20     670      PDB2:SYSAUX        NO
/u02/app/oracle/oradata/ORCL/PDB2/sysaux01.dbf
21     200      PDB2:UNDOTBS1      YES
/u02/app/oracle/oradata/ORCL/PDB2/undotbs01.dbf
38      7      PDB1:INVENTORY      NO
/u02/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF

```

List of Temporary Files

File	Size (MB)	Tablespace	Maxsize (MB)	Tempfile Name
1	130	TEMP	32767	
		/u04/app/oracle/oradata/temp/temp01.dbf		
2	62	PDB\$SEED:TEMP	32767	
		/u04/app/oracle/oradata/temp/pdbseed_temp012018-02-08_13-49-27-256-PM.dbf		
3	62	PDB2:TEMP	32767	
		/u04/app/oracle/oradata/temp/PDB2/pdbseed_temp012018-02-08_13-49-27-256-PM.dbf		
4	62	PDB1:TEMP	32767	
		/u04/app/oracle/oradata/temp/temp012018-02-08_13-49-27-256-PM.dbf		

RMAN>

3. Back up the whole database. Your results will be different from the results shown below; for example, the piece handle names will be different.

```
RMAN> BACKUP DATABASE;

Starting backup at 09-APR-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=46 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00001
name=/u02/app/oracle/oradata/ORCL/system01.dbf
input datafile file number=00003
name=/u02/app/oracle/oradata/ORCL/sysaux01.dbf
input datafile file number=00004
name=/u02/app/oracle/oradata/ORCL/undotbs01.dbf
input datafile file number=00007
name=/u02/app/oracle/oradata/ORCL/users01.dbf
channel ORA_DISK_1: starting piece 1 at 09-APR-18
channel ORA_DISK_1: finished piece 1 at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04
_09/o1_mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
tag=TAG20180409T160501 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:01:25
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00013
name=/u02/app/oracle/oradata/ORCL/PDB1/sysaux01
...
channel ORA_DISK_1: finished piece 1 at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/69350B8874FA03C8E
053A23F160AC9F7/backupset/2018_04_09/o1_mf_nnndf_TAG20180409T160
501_fdql1xjr_.bkp tag=TAG20180409T160501 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:55
Finished backup at 09-APR-18

Starting Control File and SPFILE Autobackup at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973008564_fdql13o2s_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 09-APR-18

RMAN>
```

4. Question: Do you have to shut down the database to back it up?

Answer: No, as long as the database is in ARCHIVELOG mode, the backup can take place while the database is opened. This is a hot backup (or online backup). A cold backup (or offline backup) is a backup completed while the database is closed and is required if the database is in NOARCHIVELOG mode.

5. Question: Are hot backups consistent?

Answer: Online backups are inconsistent because with the database opened, there is no guarantee that the data files are synchronized with the control files. However, offline backups taken while the database is not opened are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

6. Question: What would allow hot backups (inconsistent backups) to perform a complete database recovery?

Answer: During a complete recovery, restored online backups are recovered until the current SCN is matched, with the use of the archive log files and online redo log files.

7. Question: Did the backup include the SPFILE and control files?

Answer: Yes. This is the last operation completed at the end of the backup command.

```
...
Starting Control File and SPFILE Autobackup at 09-APR-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973008564_fdql3o2s_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 09-APR-18
```

8. Question: Does the complete operation create a single backup set?

Answer: No. The operation creates multiple backup sets.

- Four backup sets including data files (one for each of the containers): CDB root, PDB seed, PDB1, PDB2
- One backup set for the SPFILE and control files.

9. List the backup sets. Look for Piece Name in the results for each backup set.

```
RMAN> LIST BACKUP;

List of Backup Sets
=====
BS Key  Type LV Size      Device Type Elapsed Time Completion
Time
-----
--  --
2       Full   17.95M    DISK        00:00:01      05-APR-18
          BP Key: 2     Status: AVAILABLE  Compressed: NO   Tag:
TAG20180405T170539
```

```

Piece Name:
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_05/o1
_mf_s_972666339_fdf3x4bf_.bkp
SPFILE Included: Modification time: 05-APR-18
SPFILE db_unique_name: ORCL
Control File Included: Ckp SCN: 2845204 Ckp time: 05-APR-
18
...
BS Key Type LV Size Device Type Elapsed Time Completion
Time
-----
---  

10 Full 18.23M DISK 00:00:01 09-APR-18
BP Key: 10 Status: AVAILABLE Compressed: NO Tag:
TAG20180409T160924
Piece Name:
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_09/o1
_mf_s_973008564_fdql3o2s_.bkp
SPFILE Included: Modification time: 09-APR-18
SPFILE db_unique_name: ORCL
Control File Included: Ckp SCN: 3108607 Ckp time: 09-APR-
18

RMAN>

```

10. Exit RMAN.

```
RMAN> EXIT
```

11. Verify that the files are stored on disk in 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
```

```

./69350B8874FA03C8E053A23F160AC9F7:
total 4
drwxr-x--- 3 oracle oinstall 4096 Apr  9 16:08 backupset

...
./archivelog/2018_04_06:
total 0

./archivelog/2018_04_05:
total 0

./flashback:
total 2097176
-rwxr-x--- 1 oracle oinstall 1073750016 Apr  9 16:15
o1_mf_fdf30qbz_.fbz
-rwxr-x--- 1 oracle oinstall 1073750016 Apr  5 16:52
o1_mf_fdf32pm6_.fbz

./onlinelog:
total 0
[oracle@MYDBCS ORCL]$

```

- Question: Where are the backups of control files and SPFILE located?

Answer: They are created in the `autobackup` subdirectory.

- Question: How are backups deleted?

Answer: Space management in the FRA is governed by a backup retention policy. A retention policy determines when files are obsolete, which means that they are no longer needed to meet your data recovery objectives. The Oracle Database server automatically manages this storage by deleting files that are no longer needed.

12. View the backup retention policy.

- Start RMAN and connect to the CDB root as the `SYS` user.

```
$ rman target /
```

- Issue the `SHOW RETENTION POLICY` command. The policy is `REDUNDANCY 1`.

```
RMAN> SHOW RETENTION POLICY;
```

```
using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name
ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
```

```
RMAN>
```

13. Question: How does Oracle determine when files are obsolete?

Answer: There are two retention policy parameters that are mutually exclusive:

- If a retention policy is enabled with RECOVERY WINDOW OF 5 DAYS, the window stretches from the current time (SYSDATE) to the point of recoverability, which is the earliest date to which you want to recover. The point of recoverability is SYSDATE - integer days in the past.
- If a retention policy is enabled with REDUNDANCY r , then RMAN skips backups only if at least n backups of an identical file exist on the specified device, where $n=r+1$ (default is 1).

RMAN automatically deletes obsolete backup sets and copies in the FRA when space is needed.

14. Manually delete obsolete files by issuing the DELETE OBSOLETE command.

```
RMAN> delete obsolete;

using target database control file instead of recovery catalog
RMAN retention policy will be applied to the command
RMAN retention policy is set to redundancy 1
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=4 device type=DISK
Deleting the following obsolete backups and copies:
Type          Key      Completion Time    Filename/Handle
-----  -----
--- 
Backup Set      2        05-APR-18
    Backup Piece   2        05-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_05/o1
_mf_s_972666339_fdf3x4bf_.bkp
Backup Set      3        06-APR-18
    Backup Piece   3        06-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_06/o1
_mf_s_972751516_fdhq2y5d_.bkp
Backup Set      4        09-APR-18
    Backup Piece   4        09-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_09/o1
_mf_s_973004494_fdqg4gqp_.bkp
Backup Set      5        09-APR-18
    Backup Piece   5        09-APR-18
/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_04_09/o1
_mf_s_973005697_fdqhb2cd_.bkp

Do you really want to delete the above objects (enter YES or
NO)? yes
deleted backup piece
```

```

backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_05/o1_mf_s_972666339_fdf3x4bf_.bkp  RECID=2  STAMP=972666340
deleted backup piece
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_06/o1_mf_s_972751516_fdhq2y5d_.bkp  RECID=3  STAMP=972751517
deleted backup piece
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973004494_fdqg4gqp_.bkp  RECID=4  STAMP=973004494
deleted backup piece
backup piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
4_09/o1_mf_s_973005697_fdqhb2cd_.bkp  RECID=5  STAMP=973005698
Deleted 4 objects

```

RMAN>

15. Back up the database and archive logs as image copies. At the same time, free space in the FRA by deleting the archive log files once they are backed up.

- a. Perform the backup.

```

RMAN> BACKUP AS COPY DATABASE PLUS ARCHIVELOG DELETE INPUT;

Starting backup at 06-JUN-18
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=284 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=15 RECID=15 STAMP=978029619
output file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_06_
06/o1_mf_1_15_fkj1fpws_.arc RECID=21 STAMP=978105797
...
Finished backup at 06-JUN-18

Starting backup at 06-JUN-18
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
input datafile file number=00003
name=/u02/app/oracle/oradata/ORCL/sysaux01.dbf
...
Starting Control File and SPFILE Autobackup at 06-JUN-18

```

```
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0
6_06/o1_mf_s_978106455_fkj22r3p_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 06-JUN-18
```

```
RMAN>
```

- b. Question: What would you do if an error such as the following occurs?

```
RMAN-00571:
=====
RMAN-00569: ====== ERROR MESSAGE STACK FOLLOWS
=====
RMAN-00571:
=====
RMAN-03002: failure of backup plus archivelog command at
01/23/2017 11:05:08
ORA-19809: limit exceeded for recovery files
ORA-19804: cannot reclaim 67108864 bytes disk space from
19327352832 bytes limit
```

Answer: Increase the DB_RECOVERY_FILE_DEST_SIZE parameter value to 30G by issuing the following command:

```
RMAN> ALTER SYSTEM SET db_recovery_file_dest_size = 30G
SCOPE=both;
```

- c. Question: What is the advantage of creating backups as image copies?

Answer: The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy, only the file or files need to be retrieved from your backup location. With backup sets, the entire backup set must be retrieved from your backup location before you extract the file or files that are needed.

- d. Question: What is the advantage of creating backups as backup sets?

Answer: The advantage of creating backups as backup sets is better space usage. In most databases, 20% or more of the data blocks are empty blocks. Image copies back up every data block, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems, the advantages of backup sets outweigh the advantages of image copies.

- e. Question: How many image copies of the data files are created?

Answer: There are 15 image copies, one image copy for each data file in the CDB, PDBs included.

16. Exit RMAN.

```
RMAN> EXIT
```

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>
```

2. Back up PDB1, including the archived redo log files.

```
RMAN> BACKUP PLUGGABLE DATABASE PDB1 PLUS ARCHIVELOG;  
  
Starting backup at 06-JUN-18  
current log archived  
using target database control file instead of recovery catalog  
allocated channel: ORA_DISK_1  
channel ORA_DISK_1: SID=41 device type=DISK  
channel ORA_DISK_1: starting archived log backup set  
channel ORA_DISK_1: specifying archived log(s) in backup set  
input archived log thread=1 sequence=14 RECID=22 STAMP=978105812  
...  
Starting Control File and SPFILE Autobackup at 06-JUN-18  
piece  
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0  
6_06/o1_mf_s_978108946_fkj4jm29_.bkp comment=NONE  
Finished Control File and SPFILE Autobackup at 06-JUN-18  
  
RMAN>
```

3. Exit RMAN.

```
RMAN> EXIT
Recovery Manager complete.
[oracle@MYDBCS ORCL]$
```

4. Question: Did the partial backup automatically include the SPFILE and control files?

Answer: Yes. The setting verified in Practice 18-2 Verifying Automatic Backups of the Control File and SPFILE is also valid for partial backups.

5. Question: How many backup sets are created?

Answer: Four backup sets: one for the PDB data files, one for the SPFILE and control file, one for the archived log files before the data file backup set, and one for the archived log files after the data file backup set.

6. Question: Can you connect in RMAN directly to the PDB to perform the same backup?

Answer: Yes. In this case, you do not have to specify that you want to back up a PDB. Instead, you can use the simple BACKUP DATABASE command.

7. Perform a partial database backup in PDB1 directly.

a. Start RMAN and connect to PDB1 as the SYS user.

```
[oracle@MYDBCS ORCL]$ rman target SYS/password@PDB1
...
connected to target database: ORCL:PDB1 (DBID=2133829969)

RMAN>
```

b. Execute the BACKUP DATABASE command. Notice that the SPFILE and control file are not backed up.

```
RMAN> BACKUP DATABASE;
Starting backup at 06-JUN-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=41 device type=DISK
...
channel ORA_DISK_1: starting piece 1 at 06-JUN-18
channel ORA_DISK_1: finished piece 1 at 06-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6D5DA61701C1285EE
0533E89810ACAA5/backupset/2018_06_06/o1_mf_nnndf_TAG20180606T165
917_fkj4q7w6_.bkp tag=TAG20180606T165917 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:01:25
Finished backup at 06-JUN-18

RMAN>
```

- 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>
```

- Exit RMAN.

```
RMAN> EXIT
```

- Back up the TBS_APP tablespace in PDB2.

- Connect to PDB2 as the SYS user.

```
$ rman target SYS/password@PDB2
...
connected to target database: ORCL:PDB2 (DBID=3237529478)

RMAN>
```

- Back up the tablespace.

```
RMAN> BACKUP TABLESPACE tbs_app;
Starting backup at 06-JUN-18
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=55 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00059
name=/u02/app/oracle/oradata/ORCL/PDB2/tbs_app01.dbf
channel ORA_DISK_1: starting piece 1 at 06-JUN-18
channel ORA_DISK_1: finished piece 1 at 06-JUN-18
piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/6D975E8B80B85F14E
0537A051D0A3C0D/backupset/2018_06_06/o1_mf_nnndf_TAG20180606T170
407_fkj5091o_.bkp tag=TAG20180606T170407 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:04
Finished backup at 06-JUN-18
```

```
RMAN>
```

- c. Exit RMAN.

```
RMAN> EXIT
```

```
Recovery Manager complete.  
[oracle@DKKDBCS ORCL] $
```

11. Can you connect to the CDB root and perform the same operation?

- a. Start RMAN and connect to the CDB root as the SYS user.

```
$ rman target /
```

```
...
```

```
connected to target database: ORCL (DBID=1505229725)
```

```
RMAN>
```

- b. Back up the TBS_APP tablespace in PDB2. You must specify the PDB in which the tablespace exists.

```
RMAN> BACKUP TABLESPACE PDB2:tbs_app;
```

```
Starting backup at 06-JUN-18
```

```
using target database control file instead of recovery catalog  
allocated channel: ORA_DISK_1
```

```
channel ORA_DISK_1: SID=271 device type=DISK
```

```
channel ORA_DISK_1: starting full datafile backup set
```

```
channel ORA_DISK_1: specifying datafile(s) in backup set
```

```
input datafile file number=00059
```

```
name=/u02/app/oracle/oradata/ORCL/PDB2/tbs_app01.dbf
```

```
channel ORA_DISK_1: starting piece 1 at 06-JUN-18
```

```
channel ORA_DISK_1: finished piece 1 at 06-JUN-18
```

```
piece
```

```
handle=/u03/app/oracle/fast_recovery_area/ORCL/6D975E8B80B85F14E  
0537A051D0A3C0D/backupset/2018_06_06/o1_mf_nnndf_TAG20180606T170  
547_fkj53f1h_.bkp tag=TAG20180606T170547 comment=NONE
```

```
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
```

```
Finished backup at 06-JUN-18
```

```
Starting Control File and SPFILE Autobackup at 06-JUN-18
```

```
piece
```

```
handle=/u03/app/oracle/fast_recovery_area/ORCL/autobackup/2018_0  
6_06/o1_mf_s_978109551_fkj53l2p_.bkp comment=NONE
```

```
Finished Control File and SPFILE Autobackup at 06-JUN-18
```

```
RMAN>
```

- c. Question: Did the operation back up only the tablespace data files?

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.

- d. 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 ~]$
```

2. Source the `oraenv` script.

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

3. 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 ~]$
```

4. Attempt an administrative task, such as creating a user.
 - a. 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>
```

- b. 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>
```

- c. Exit SQL*Plus.

```
SQL> EXIT
...

```

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

5. Consider your recovery options.

- a. 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).

- b. Question: Which methods can you use to recover the data file?

- c. 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>
```

2. Issue the RESTORE command. You must provide the number for the missing data file.

```
RMAN> RESTORE DATAFILE 1;  
  
Starting restore at 13-JUN-18  
using target database control file instead of recovery catalog  
allocated channel: ORA_DISK_1  
channel ORA_DISK_1: SID=46 device type=DISK  
  
channel ORA_DISK_1: restoring datafile 00001  
input datafile copy RECID=2 STAMP=978678712 file  
name=/u03/app/oracle/fast_recovery_area/ORCL/datafile/o1_mf_syst  
em_f11jwb73_.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
11jwb73_.dbf)
ORA-19601: output file is datafile 1
(/u02/app/oracle/oradata/ORCL/system01.dbf)
RMAN>
```

3. Question: What does the error message "cannot obtain exclusive enqueue for datafile 1" mean?

Answer: The restore operation requires an exclusive enqueue lock on the data file 1 that is not obtainable. In this case, you have to open the database instance in `MOUNT` mode. This means that you have to shut down the database instance if it did not already abort.

4. Question: What does a database instance shut down do?

Answer: This closes all PDBs and therefore prevents all users from working during the recovery operation.

5. Try to shut down the database instance in `IMMEDIATE` mode. You get an error.

```
RMAN> SHUTDOWN IMMEDIATE

RMAN-00571:
=====
RMAN-00569: ====== ERROR MESSAGE STACK FOLLOWS
=====
RMAN-00571:
=====
RMAN-03002: failure of shutdown command at 06/13/2018 08:10:09
ORA-01116: error in opening database file 1
ORA-01110: data file 1:
  '/u02/app/oracle/oradata/ORCL/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3

RMAN>
```

6. Question: Why does the `SHUTDOWN IMMEDIATE` command fail?

Answer: RMAN needs to close all data files cleanly by writing the current SCN to all data file headers. This cannot be done because data file 1 is missing.

7. Reconnect to the database.

8. Try to shut down the database instance in `ABORT` mode.

```
RMAN> SHUTDOWN ABORT

Oracle instance shut down
```

```
RMAN>
```

9. Start the database instance in MOUNT mode.

```
RMAN> STARTUP MOUNT
```

```
Oracle instance started  
database mounted
```

```
Total System Global Area    2768239832 bytes
```

```
Fixed Size                  8899800 bytes  
Variable Size               704643072 bytes  
Database Buffers           1979711488 bytes  
Redo Buffers                74985472 bytes
```

```
RMAN>
```

10. Restore the missing data file.

```
RMAN> RESTORE DATAFILE 1;
```

```
Starting restore at 10-APR-18  
allocated channel: ORA_DISK_1  
channel ORA_DISK_1: SID=25 device type=DISK  
  
channel ORA_DISK_1: starting datafile backup set restore  
channel ORA_DISK_1: specifying datafile(s) to restore from  
backup set  
channel ORA_DISK_1: restoring datafile 00001 to  
/u02/app/oracle/oradata/ORCL/system01.dbf  
channel ORA_DISK_1: reading from backup piece  
/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04_09/o1_  
mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp  
channel ORA_DISK_1: piece  
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04  
_09/o1_mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp  
tag=TAG20180409T160501  
channel ORA_DISK_1: restored backup piece 1  
channel ORA_DISK_1: restore complete, elapsed time: 00:00:15  
Finished restore at 10-APR-18
```

```
RMAN>
```

11. Recover the missing data file.

```
RMAN> RECOVER DATAFILE 1;
```

```
Starting recover at 10-APR-18
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 14 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1
_mf_1_14_fdskgq34_.arc
archived log for thread 1 with sequence 15 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1
_mf_1_15_fdsmhyob_.arc
archived log for thread 1 with sequence 16 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1
_mf_1_16_fdsmhyqz_.arc
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_
10/o1_mf_1_14_fdskgq34_.arc thread=1 sequence=14
media recovery complete, elapsed time: 00:00:01
Finished recover at 10-APR-18

RMAN>
```

12. Open the CDB root.

```
RMAN> ALTER DATABASE OPEN;

Statement processed

RMAN>
```

13. Open all PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;

Statement processed

RMAN>
```

14. Exit RMAN.

```
RMAN> EXIT

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

15. Start SQL*Plus and connect to the CDB root as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
$ sqlplus SYSTEM/password
...
SQL>
```

16. 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>
```

17. Keep Window 1 open for the next section.

Use the Data Recovery Advisor to Recover the Database

1. **Window 2:** Open a new terminal window and execute the RMAN_crash.sh script to create a failure. This window will be referred to as Window 2.

```
$ $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 ~]$
```

2. **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>
```

3. 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.

4. **Window 1:** Attempt to resize datafile 1. If it completes, execute the ALTER SYSTEM SWITCH LOGFILE command. You should receive an error message about the missing data file.

```
SQL> ALTER DATABASE DATAFILE 1 RESIZE 1G;
ALTER DATABASE DATAFILE 1 RESIZE 1G
*
ERROR at line 1:
ORA-01565: error in identifying file
'/u02/app/oracle/oradata/ORCL/system01.dbf'
ORA-27037: unable to obtain file status
Linux-x86_64 Error: 2: No such file or directory
Additional information: 7

SQL>
```

5. **Window 1:** Exit SQL*Plus.

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

6. **Window 1:** Start RMAN and connect to the target database.

```
$ rman target /
...
connected to target database (not started)

RMAN>
```

7. **Window 1:** Start the database instance in MOUNT mode.

```
RMAN> STARTUP MOUNT;

Oracle instance started
database mounted

Total System Global Area    2768239832 bytes

Fixed Size                  8899800  bytes
Variable Size                704643072  bytes
Database Buffers            1979711488  bytes
Redo Buffers                 74985472  bytes

RMAN>
```

8. **Window 1:** Use the LIST FAILURE command to determine the error. The value in the Summary column tells you that system01.dbf is missing.

```
RMAN> LIST FAILURE;

using target database control file instead of recovery catalog
Database Role: PRIMARY

List of Database Failures
=====

Failure ID Priority Status      Time Detected Summary
----- ----- -----
262       CRITICAL OPEN      10-APR-18      System datafile 1:
'/u02/app/oracle/oradata/ORCL/system01.dbf' is missing

RMAN>
```

9. **Window 1:** Display repair options. At the very end of the results, a repair script is listed.

```
RMAN> ADVISE FAILURE;

Database Role: PRIMARY

List of Database Failures
=====

Failure ID Priority Status      Time   Detected Summary
----- ----- -----
262       CRITICAL OPEN      10-APR-18      System datafile 1:
'/u02/app/oracle/oradata/ORCL/system01.dbf' is missing

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=25 device type=DISK
analyzing automatic repair options complete

Mandatory Manual Actions
=====
no manual actions available

Optional Manual Actions
=====
1. If file /u02/app/oracle/oradata/ORCL/system01.dbf was
unintentionally renamed or moved, restore it
```

```
Automated Repair Options
=====
Option Repair Description
-----
1      Restore and recover datafile 1
      Strategy: The repair includes complete media recovery with no
      data loss
      Repair script:
      /u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2771153169.hm

RMAN>
```

10. **Window 1:** Use the REPAIR FAILURE PREVIEW command to generate a script with all repair actions and comments.

```
RMAN> REPAIR FAILURE PREVIEW;

Strategy: The repair includes complete media recovery with no
data loss
Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2771153169.hm

contents of repair script:
# restore and recover datafile
restore ( datafile 1 );
recover datafile 1;
sql 'alter database datafile 1 online';

RMAN>
```

11. **Window 1:** Use the REPAIR FAILURE command to repair database failures identified by the Data Recovery Advisor. When prompted, enter YES to execute the repair. When prompted to open the database, enter YES.

```
RMAN> REPAIR FAILURE;

Strategy: The repair includes complete media recovery with no
data loss
Repair script:
/u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_2771153169.hm

contents of repair script:
# restore and recover datafile
restore ( datafile 1 );
recover datafile 1;
sql 'alter database datafile 1 online';
```

```
Do you really want to execute the above repair (enter YES or
NO) ? YES
executing repair script

Starting restore at 10-APR-18
using channel ORA_DISK_1

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from
backup set
channel ORA_DISK_1: restoring datafile 00001 to
/u02/app/oracle/oradata/ORCL/system01.dbf
channel ORA_DISK_1: reading from backup piece
/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04_09/o1_
mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
channel ORA_DISK_1: piece
handle=/u03/app/oracle/fast_recovery_area/ORCL/backupset/2018_04_
09/o1_mf_nnndf_TAG20180409T160501_fdqkvh9t_.bkp
tag=TAG20180409T160501
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:07
Finished restore at 10-APR-18

Starting recover at 10-APR-18
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 14 is already on disk as
file
/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_10/o1_
mf_1_14_fdskgq34_.arc
archived log for thread 1 with sequence 15 is already on disk as
file
...
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_
10/o1_mf_1_16_fdsmyqz_.arc thread=1 sequence=16
archived log file
name=/u03/app/oracle/fast_recovery_area/ORCL/archivelog/2018_04_
10/o1_mf_1_17_fdsn3sf0_.arc thread=1 sequence=17
media recovery complete, elapsed time: 00:00:01
Finished recover at 10-APR-18

sql statement: alter database datafile 1 online
```

```
repair failure complete

Do you want to open the database (enter YES or NO)? YES
database opened

RMAN>
```

12. **Window 1:** Open all the PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;

Statement processed

RMAN>
```

13. **Window 1:** Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.

[oracle@MYDBCS ~]$
```

14. **Window 2:** Close the terminal window.

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 ~]$
```

2. Start RMAN and connect to the CDB root as the `SYS` user.

```
$ rman target /
...
connected to target database: ORCL (DBID=1500451933)

RMAN>
```

3. 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>

```

4. 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 ~]$

```

2. Create an application table and insert data into it.

a. Connect to the CDB root as the SYS user with the SYSDBA privilege.

```

$ sqlplus / AS SYSDBA
...
SQL>

```

b. Connect to PDB1 as the SYS user.

```

SQL> ALTER SESSION SET CONTAINER = pdb1;

Session altered.

```

```
SQL>
```

- c. Create a table named OE.TEST.

```
SQL> CREATE TABLE oe.test (c NUMBER);
```

```
Table created.
```

```
SQL>
```

- d. 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>
```

3. 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.

4. 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>
```

5. Check if the TBS_APP tablespace is still online. The results indicate that it is online.

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

TABLESPACE_NAME	STATUS
SYSTEM	ONLINE
SYSAUX	ONLINE
...	
TBS_APP	ONLINE

```
SQL>
```

6. Check if PDB1 is still open. The result indicates that PDB1 is still open.

```
SQL> SHOW PDBS
```

CON_ID	CON_NAME	OPEN	MODE	RESTRICTED
4	PDB1	READ	WRITE	NO

```
SQL>
```

7. 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).

2. 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.

3. 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>
```

4. 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
-----  -----
12    350        SYSTEM            NO
/u02/app/oracle/oradata/ORCL/PDB1/system01.dbf
13    640        SYSAUX           NO
/u02/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf
14    200        UNDOTBS1         NO
/u02/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf
15    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>
```

5. 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>
```

6. 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>
```

7. 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>
```

8. Start PDB1.

```
RMAN> STARTUP

database opened

RMAN>
```

9. 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>
```

10. Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.

[oracle@MYDBCS ~]$
```

11. Try again to insert data into the `OE.TEST` table in `PDB1`.

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

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

- b. Connect to `PDB1` as the `SYS` user.

```
SQL> ALTER SESSION SET CONTAINER = pdb1;

Session altered.

SQL>
```

- c. Try to insert rows into the table.

```
SQL> INSERT INTO oe.test VALUES (1);  
  
1 row created.  
  
SQL>
```

12. 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 ~] $
```

2. Create an application table and try to insert data into it.

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

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

- b. Connect to PDB1 as the `SYS` user.

```
SQL> ALTER SESSION SET CONTAINER = pdb1;  
  
Session altered.  
  
SQL>
```

- c. Show the open mode of PDB1. The result shows that PDB1 is still open.

```
SQL> SHOW PDBS  
  
CON_ID CON_NAME          OPEN MODE RESTRICTED  
-----  
        4 PDB1             READ WRITE NO  
SQL>
```

- d. Create a table named OE.TEST2.

```
SQL> CREATE TABLE oe.test2 (c NUMBER);
```

```
Table created.
```

```
SQL>
```

- e. 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>
```

- f. 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.

3. Exit SQL*Plus.

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

4. Start RMAN and log in to the target database as the SYS user.

```
$ rman target /
...
connected to target database: ORCL (DBID=1500451933)

RMAN>
```

5. 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>
```

6. Open PDB1.

```
RMAN> ALTER PLUGGABLE DATABASE pdb1 OPEN;  
  
Statement processed  
  
RMAN>
```

7. Exit RMAN.

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

8. Try to insert data into the OE.TEST2 table again.

- Start SQL*Plus and connect to PDB1 as the SYSTEM user.

```
$ sqlplus SYSTEM/password@PDB1  
...  
SQL>
```

- 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>
```

- Commit the transaction.

```
SQL> COMMIT;  
  
Commit complete.  
  
SQL>
```

- 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 ~]$
```

2. Source the `oraenv` script.

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

3. 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 ~]$
```

4. Start an application workload in `PDB1` and `PDB2`.

```
$ $HOME/labs/PERF_loop.sh
```

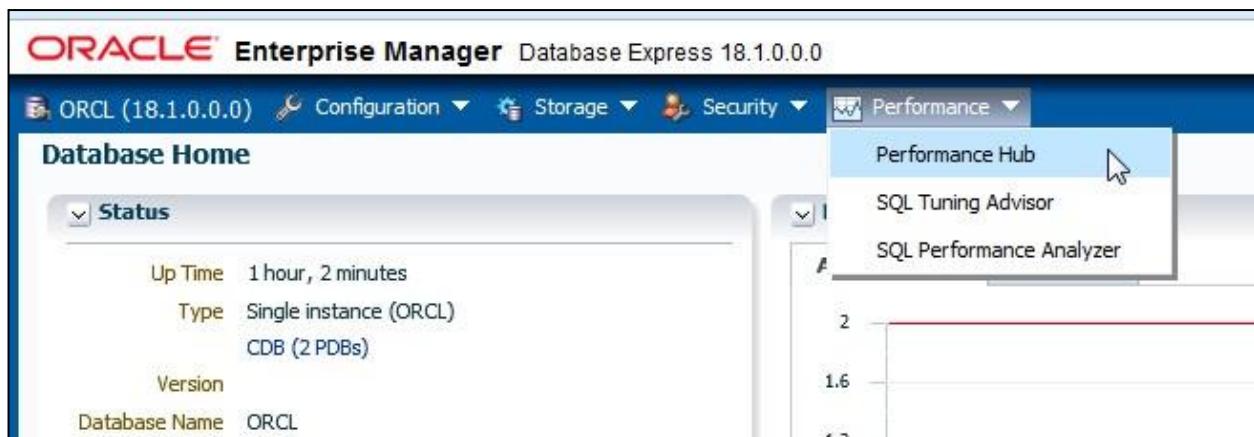
Note: This script generates continuous output in the terminal window where it starts.

Review the Performance Hub in EM Express

1. Open a new terminal window; create an SSH tunnel.

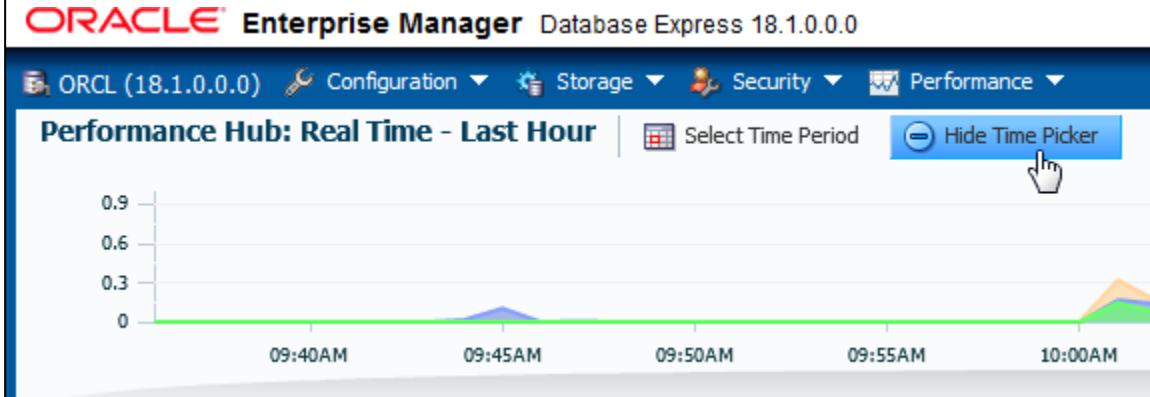
```
[oracle@edvm ~]$ cd ~/.ssh  
[oracle@edvm ~]$ ssh -i your_private_key_file -L  
5500:your_compute_node_IP_address:5500  
oracle@your_compute_node_IP_address  
[oracle@MYDBCS ~]$
```

2. Open a browser. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>.
3. On the Login page, enter the user name `SYS` and the password. Leave the Container Name box empty, select `as sysdba`, and click **Login**.
4. Select **Performance** and then **Performance Hub**.

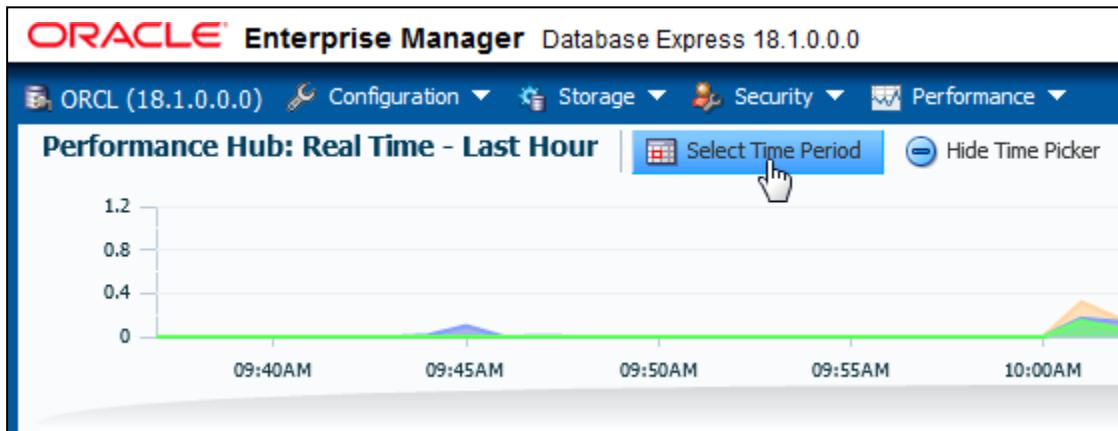


The Performance Hub provides a consolidated view of all performance data for a given time range. You must have the Oracle Diagnostics Pack (licensed option) to use the Performance Hub.

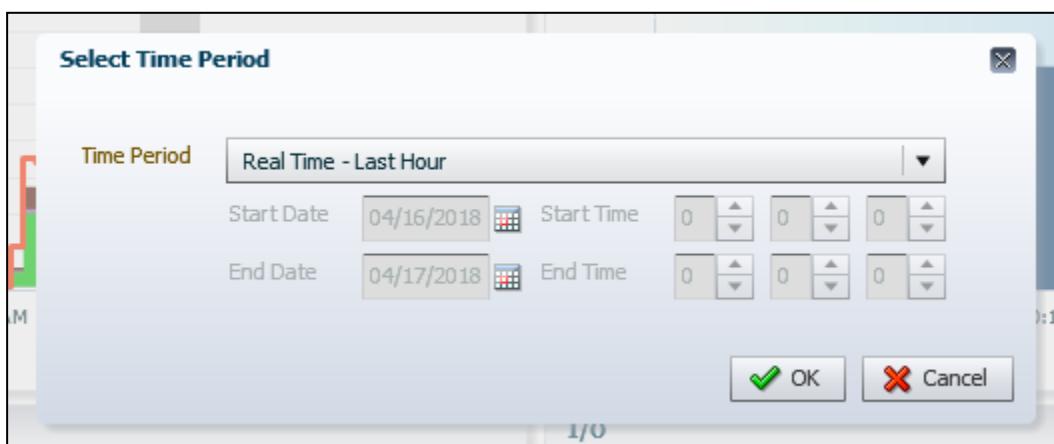
5. Learn about the Time Picker at the top of the page. The Time Picker displays average active sessions over time.
 - a. Click **Hide Time Picker** and then **Show Time Picker** to hide and show it.



- b. At the top of the page, click **Select Time Period**.

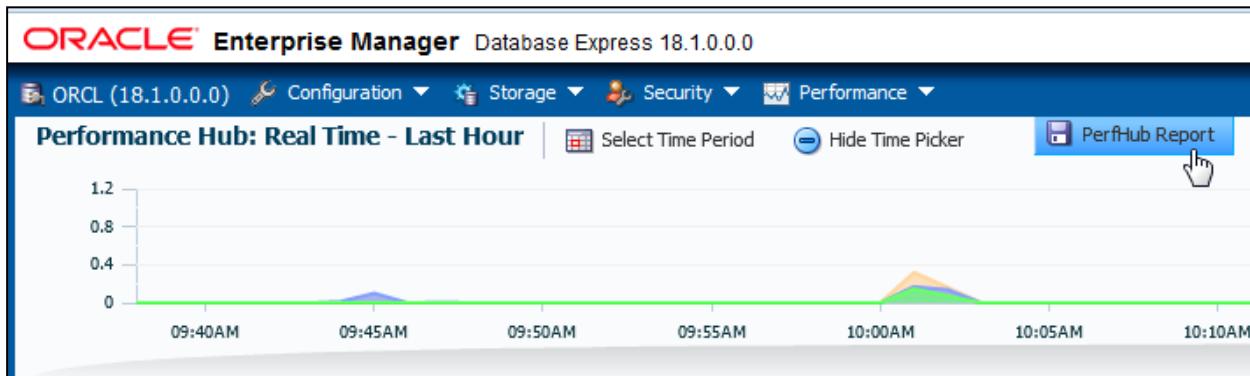


- c. In the dialog box, you can select a time range, and the detail tabs will display the available performance data for the selected time range. Click the drop-down list and review the options. Notice that you can choose to view historical and real-time data.
d. Select **Real Time - Last Hour** and 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.



- e. 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.
f. 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.
g. Change the time picker back to displaying the lasthour.

- h. Click **PerfHubReport** at the top of the page.



Notice that you save the contents of the tabs with details for top SQL statements. Click Cancel.

6. The Performance Hub organizes performance data by dividing it into different tabs. Each tab addresses a specific aspect of database performance.
- 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.
 - Click the **Activity** tab. This tab displays Active Session History (ASH) analytics and is available in both real-time and historical mode.
 - 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.
 - 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.
 - 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.
 - 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.

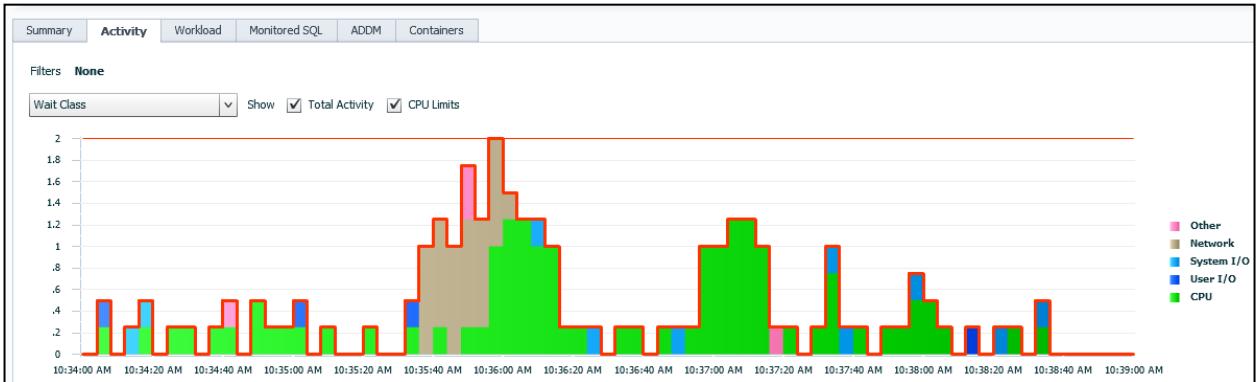
View Wait Statistics on the Activity Tab

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.

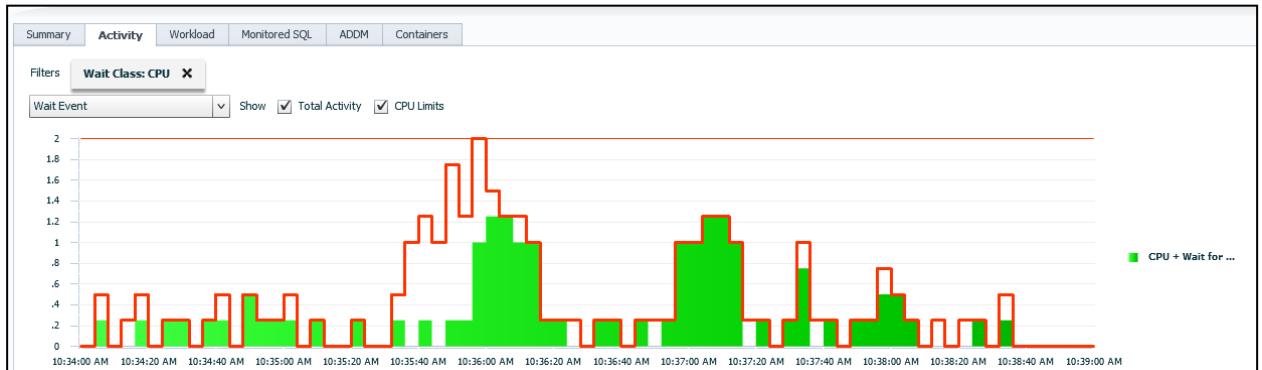
- If your workload script has ended, start it again in the terminal window.

```
$ $HOME/labs/PERF_loop.sh
```

- In EM Express, click the Refresh button a few times. Eventually you will get some data in the time picker.
- Click the **Activity** tab.



- 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.
 - 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.
 - 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.
 - 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.

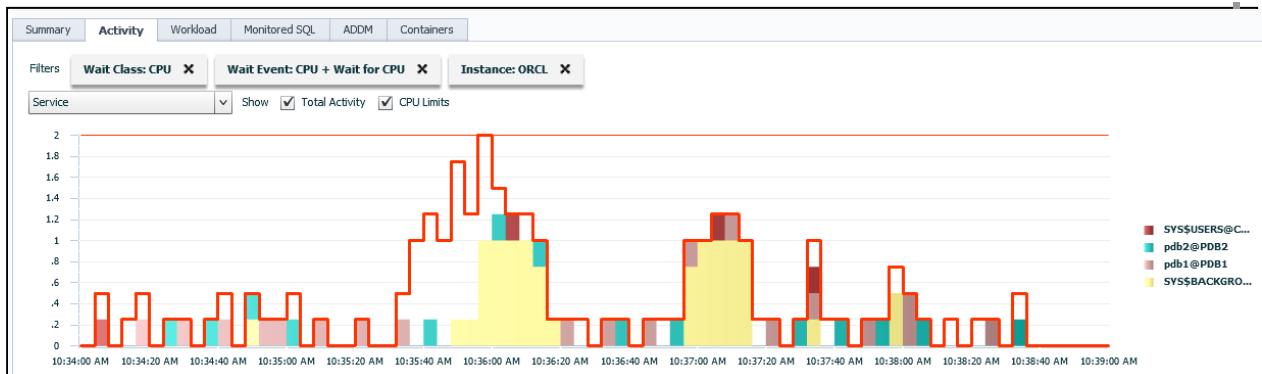


- Click the graph. The graph is displaying only one item at the moment, so don't worry about clicking the wrong part. Notice when you positioned your cursor over the graph it

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.

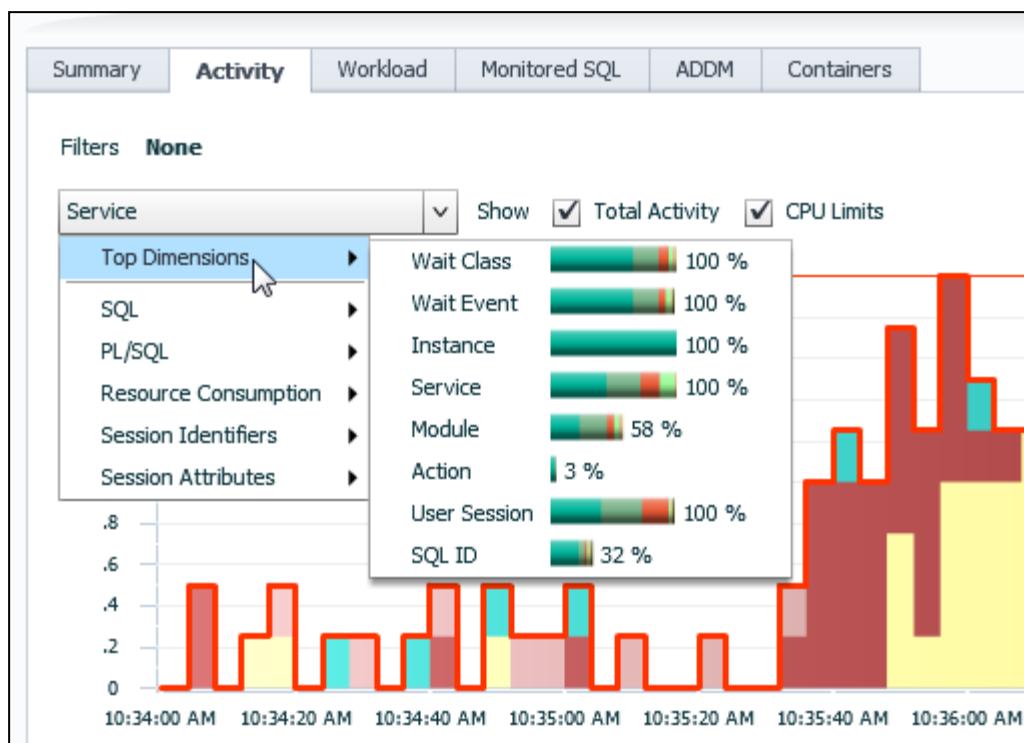


- In the legend, click **ORCL**. You have now drilled down into the waits per service.



- Remove the filters in the graph by clicking the Xs for each filter.

- g. 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.

- h. Position your cursor over **Service** and view the percentage breakdown for service waits.

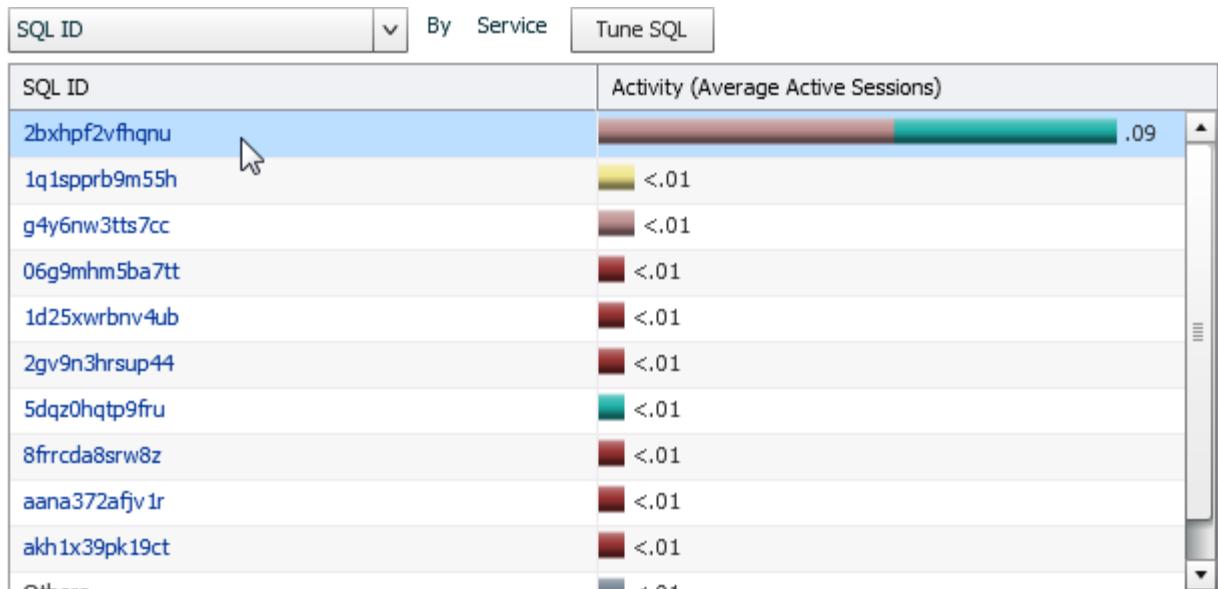
Filter Wait Statistics for a SQL ID

5. If your workload script has ended, start it again in the terminal window.

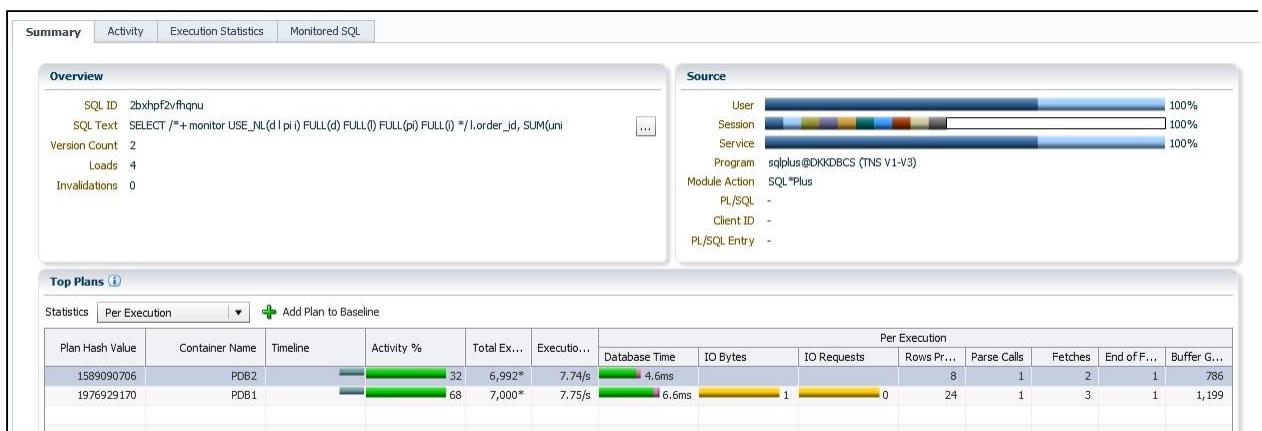
```
$ $HOME/labs/PERF_loop.sh
```

6. In EM Express, at the bottom left, view the table that shows the activity (average active sessions) for each SQLID.

- a. Click the SQL ID that has the greatest average. In this example, the SQL IDs are 2bxhpf2vfhqnu.

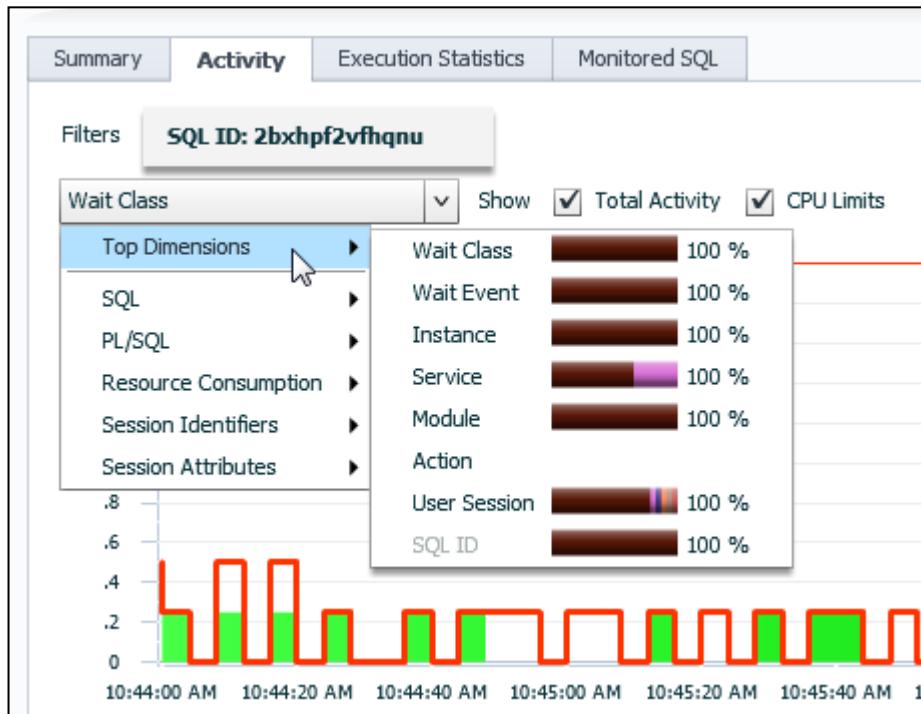


- b. The Summary tab is displayed with performance information about the selected SQL ID.

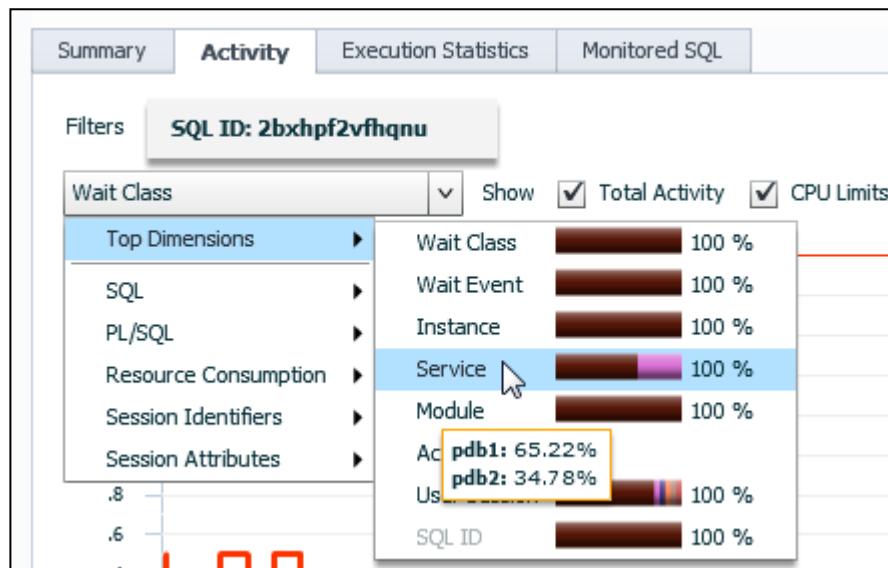


7. Click the **Activity** tab. Notice that the Activity tab is filtered based on your selected SQL ID.

8. In the filters drop-down list, select **Wait Class** and then **Top Dimensions**.



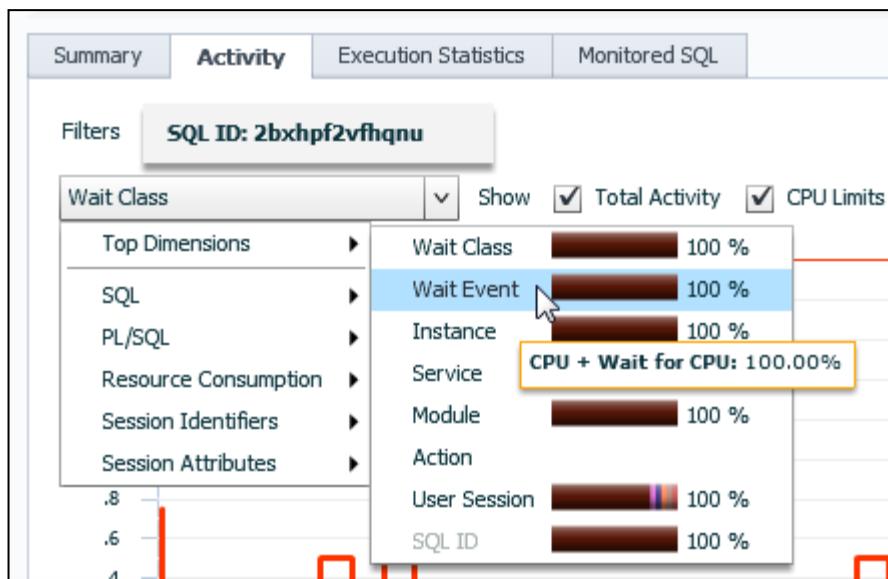
9. Position your cursor over **Service**.



10. 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.

11. Position your cursor over **Wait Event**. Notice that the execution of the statement is waiting for CPU resources.



12. Click **Log Out** to log out of EM Express, and close the browser window.

13. In the terminal window, press **Ctrl+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 ~]$
```

2. 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>
```

3. 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 ~]$
```

2. 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>
```

3. 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.

4. 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.

5. 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.

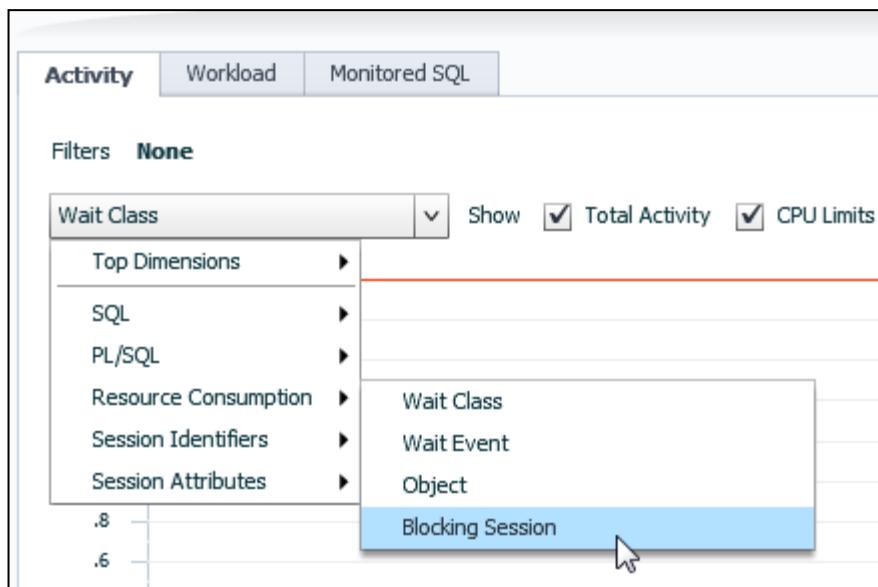
Use Enterprise Manager Database Express to Release the Lock

In this section, you use EM Express to locate the blocking session and retrieve the identifiers that enable you to kill the blocking session.

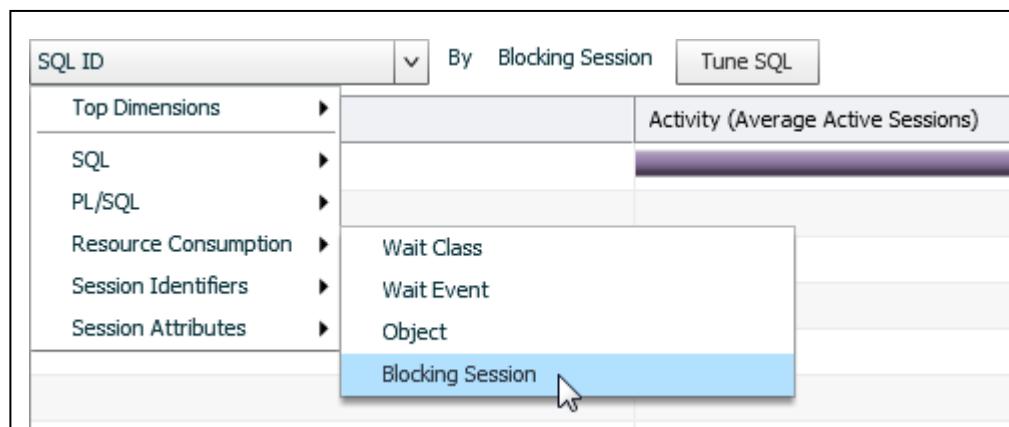
1. Open a browser. Launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
2. On the Login page for EM Express, enter the user name **sys** and the password. Enter **PDB1** as the container name. Select **as sysdba**. Click **Login**.
3. Select **Performance** and then **Performance Hub**.



4. The Activity tab is displayed by default. In the filter drop-down list, select Resource Consumption and then Blocking Session.



5. 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.



6. 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.

Blocking Session
Blocking Session
1,305,35109

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.

7. 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 ~]$
```

2. 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>
```

3. 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>
```

4. Log out of SQL*Plus and close the Window 3 terminal window.

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

Window 2

5. In Window 2, find out what happened to the blocked session. Verify that the salary was updated.
 - a. Notice that in window 2, the blocked session is now released and the result is "1 row updated."

```
1 row updated.  
  
SQL>
```

- b. 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;
```

PHONE_NUMBER	SALARY
515.124.4269	8300

SQL>

- Roll back the salary update because you don't need to keep the update.

SQL> **ROLLBACK;**

Rollback complete.

SQL>

- Log out of SQL*Plus and close the Window 2 terminal window.

SQL> **exit**

...

[oracle@MYDBCS ~]\$ **exit**

Window 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.

- 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 ~]$
```

2. Source the `oraenv` script.

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

3. 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 ~]$
```

4. 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
```

```
...
```

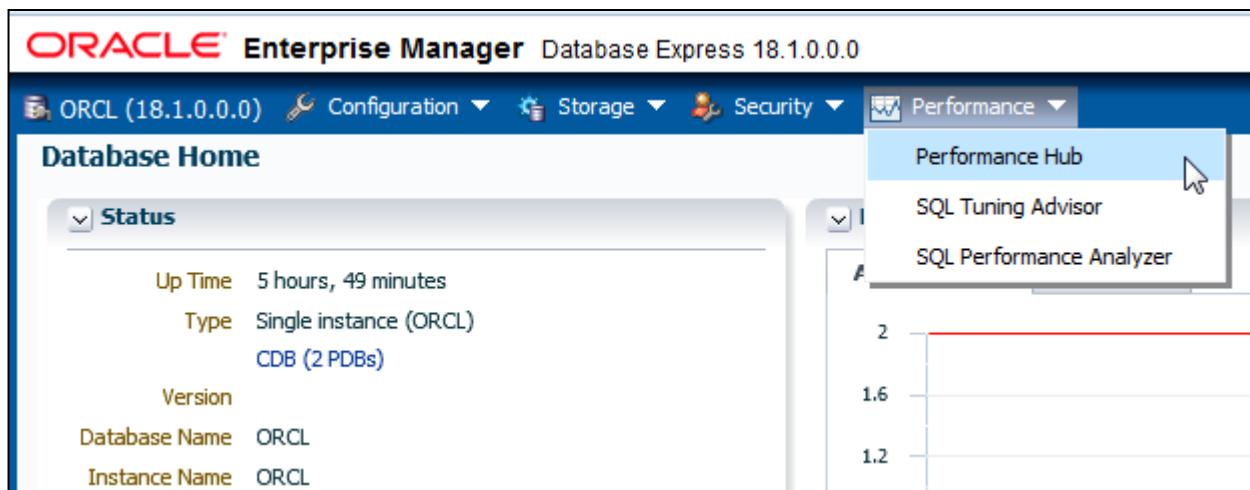
Use EM Express to Tune the SQL Based on Statistics

In this section, you review but do not implement the first recommendation of the SQL Tuning Advisor, which is based on statistics.

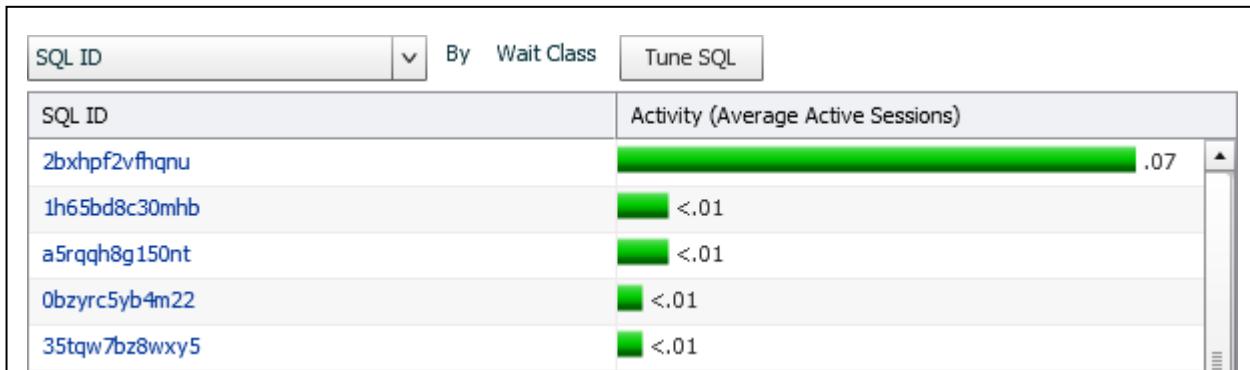
1. Open a new terminal window and create an SSH tunnel.

```
[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 ~] $
```

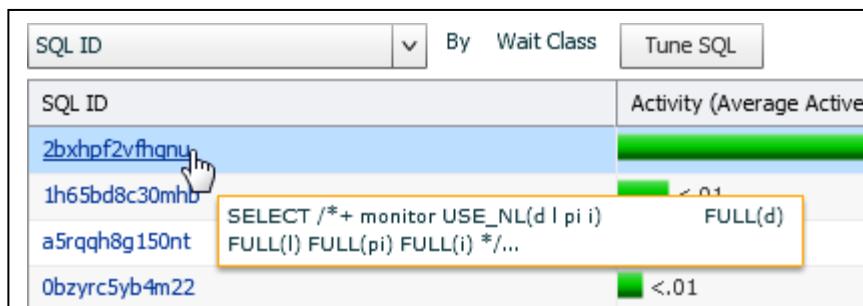
2. Open a browser and launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em>
3. On the Login page, enter the user name `sys` and the password. Leave the Container Name box empty, select `as sysdba`, and click **Login**.
4. Select **Performance** and then **Performance Hub**.



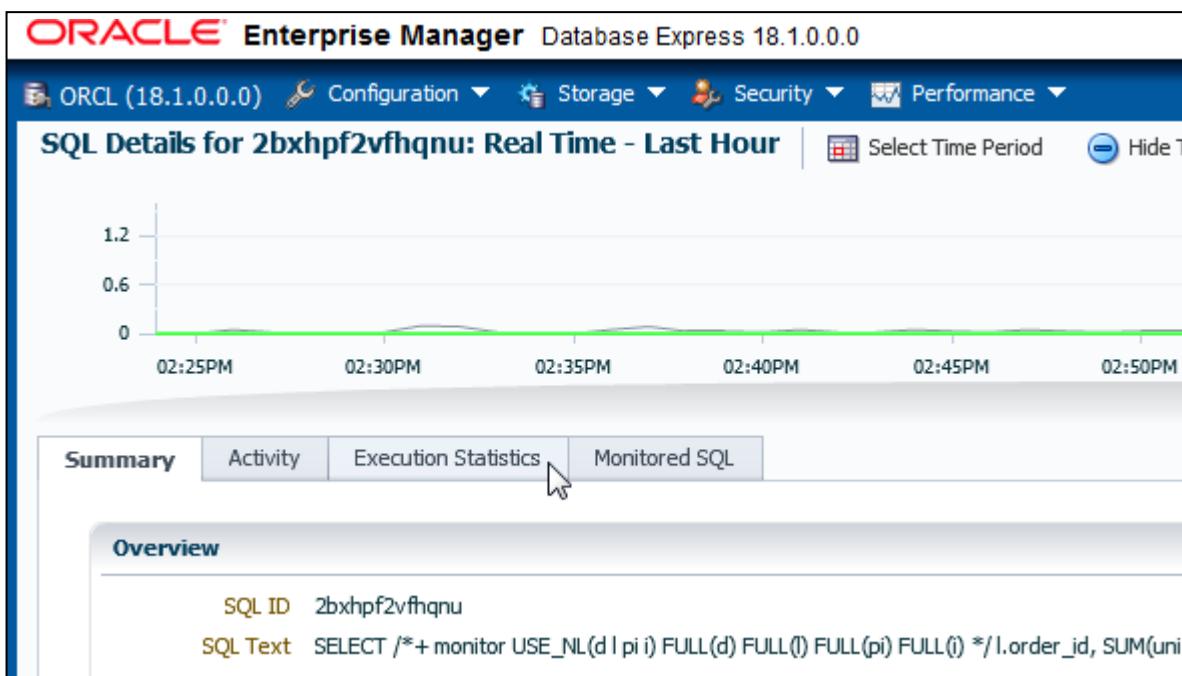
5. Click the **Activity** tab. The current SQL executions are listed in the table at the bottom of the page. In this example, you can see that there is one consuming SQL execution (the first SQL ID in the list).



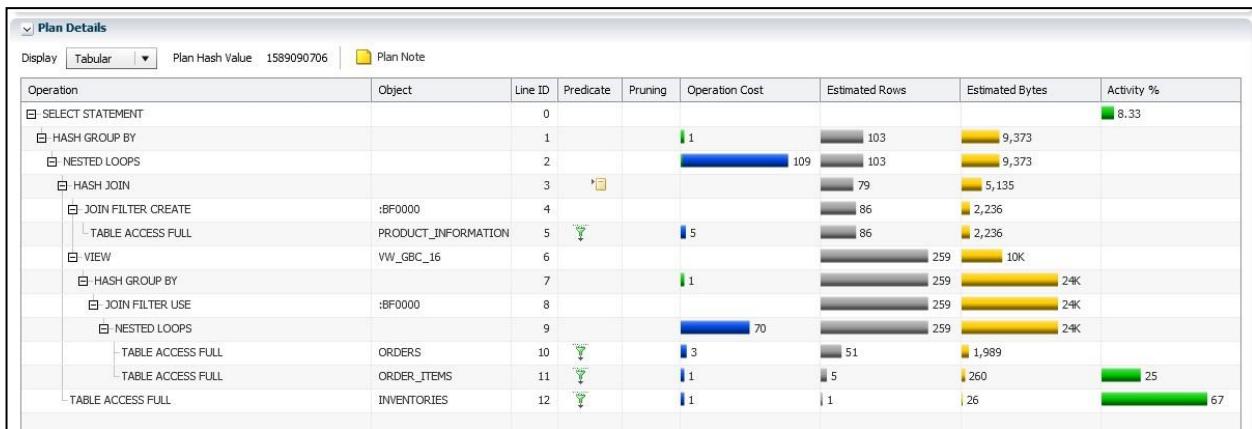
6. Position your cursor over the SQL ID. The following code should appear. If your result looks different, wait a moment and refresh EM Express.



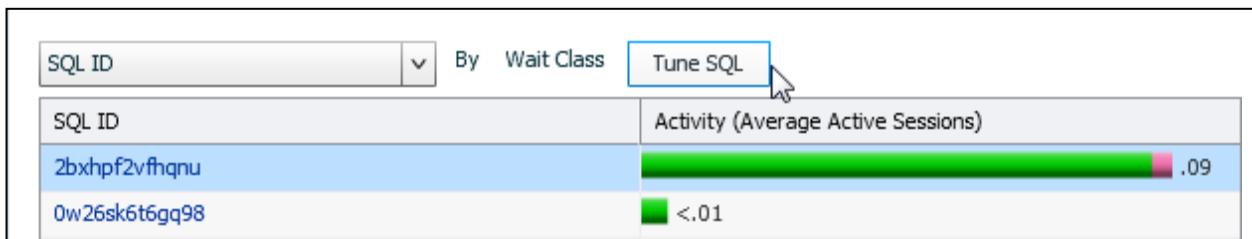
7. Click the SQL ID, and then click the **Execution Statistics** tab.



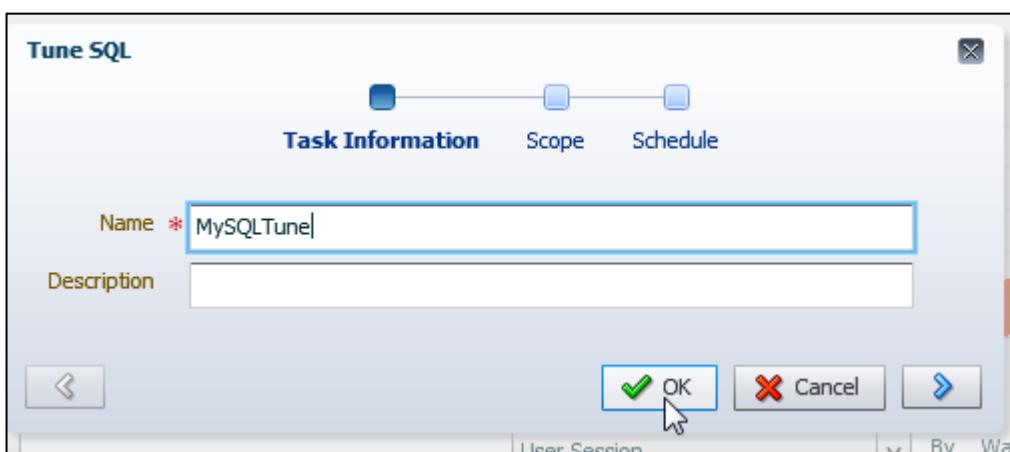
8. The Plan Details area at the bottom of the page shows you the current plan for executing the SQL.



9. To return to where you were last, select **Performance** and then **Performance Hub**. Click the **Activity** tab.
 10. 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.



11. 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.
 12. In the Tune SQL dialog box, enter the task name **MySQLTune** and click **OK**.



13. 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.

14. Click the **Manual** tab. The task you just created is listed. You may need to wait a moment for it to complete its processing.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads "ORACLE Enterprise Manager Database Express 18.1.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation is a sub-menu for "SQL Tuning Advisor" with options for "Automatic Runs" and "Show All". A "Configuration" link is also present. The main content area is titled "SQL Tuning Advisor" and contains a message about the advisor's function. Below this is a table with two tabs: "Automatic" and "Manual". The "Manual" tab is selected. The table header includes "Actions", "View Result", and "Drop". The table has columns for "Status", "Name", "Description", and "User". One row is visible, showing a checkmark icon in the "Status" column, the name "MySQLTune" in the "Name" column, and "SYS" in the "User" column.

15. Click your task to read the recommendations.
16. In the SQL Text area, you can view the full SQL statement to be tuned. Scroll down to view all of it.

The screenshot shows a "SQL Text" section with a dropdown menu. The dropdown is open, showing the option "SQL Text". Below the dropdown is a large block of SQL code. The code is a complex SELECT statement with multiple joins between tables "oe.orders" (d), "oe.order_items" (l), "oe.product_information" (pi), and "oe.inventories" (i). It includes several WHERE clauses with AND conditions, such as matching order IDs, product IDs, and date ranges. There are also clauses for quantity and price ranges.

```
SELECT /*+ monitor USE_NL(d l pi i)
          FULL(d) FULL(l) FULL(pi) FULL(i) */
          l.order_id, SUM(unit_price * quantity) amount
FROM oe.orders d , oe.order_items l,
      oe.product_information pi, oe.inventories i
WHERE d.order_id = l.order_id
AND pi.product_id = l.product_id
AND pi.product_id = i.product_id
AND d.order_date < to_DATE('10-12-2021','DD-MM-YYYY')
AND d.order_total BETWEEN 1394 AND 100000
AND l.quantity between 10 AND 300
AND pi.min_price between 100 and 500
AND i.QUANTITY_ON_HAND > 100
```

17. 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.

The screenshot shows the 'Select Recommendation' section of the Oracle Enterprise Manager interface. It displays a table with two rows: 'Statistics' and 'SQL Profile'. The 'Statistics' row contains four findings related to tables 'ORDER_ITEMS', 'INVENTORIES', 'PRODUCT_INFORMATION', and 'ORDERS' which were not analyzed. The 'SQL Profile' row indicates a potentially better execution plan was found for the statement.

Type	Findings
Statistics	Optimizer statistics are not up-to-date for objects referenced in the SQL statement. Table "OE"."ORDER_ITEMS" was not analyzed. Table "OE"."INVENTORIES" was not analyzed. Table "OE"."PRODUCT_INFORMATION" was not analyzed. Table "OE"."ORDERS" was not analyzed.
SQL Profile	A potentially better execution plan was found for this statement.

18. 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.
19. In the **Select Recommendations** area, click the **Statistics** link.
20. 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.

The screenshot shows the 'Recommendation Details' page for 'Stale or Missing Statistics'. It includes a summary message about the optimizer relying on object statistics and a list of recommendations for tables 'PRODUCT_INFORMATION', 'INVENTORIES', 'ORDER_ITEMS', and 'ORDERS' that were not analyzed.

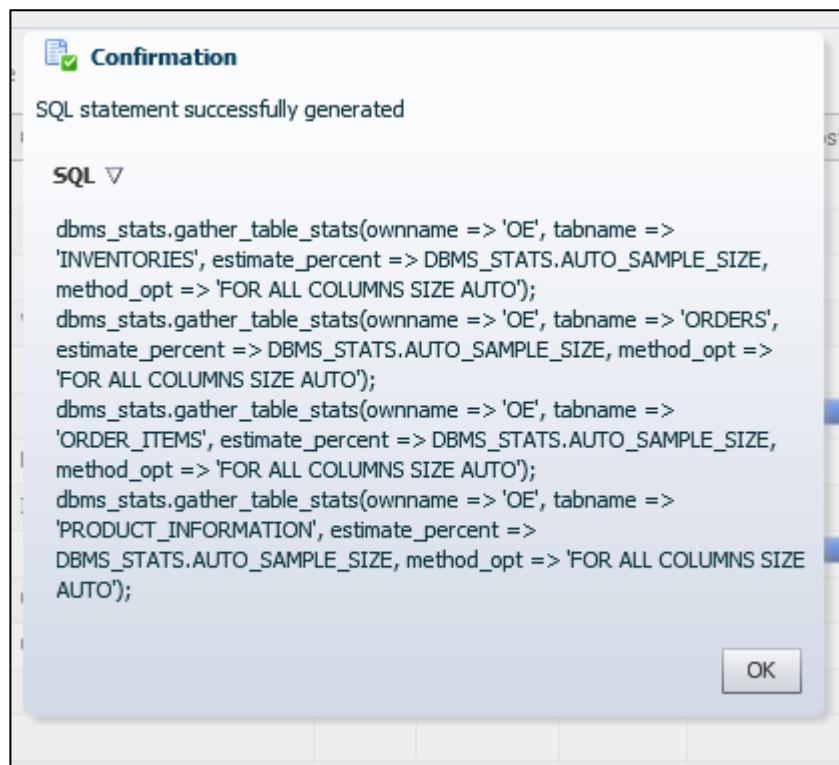
Recommendation(s):

- Table "OE"."PRODUCT_INFORMATION" was not analyzed. Consider collecting optimizer statistics for this table.
- Table "OE"."INVENTORIES" was not analyzed. Consider collecting optimizer statistics for this table.
- Table "OE"."ORDER_ITEMS" was not analyzed. Consider collecting optimizer statistics for this table.
- Table "OE"."ORDERS" was not analyzed. Consider collecting optimizer statistics for this table.

21. In the Gather Statistics dialog box, click **Show SQL**.



22. View the SQL package and procedure that would gather the statistics, and click **OK**.



23. 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**.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The top navigation bar has tabs for Configuration, Storage, Security, and Performance. The 'Performance' tab is currently selected. A dropdown menu under 'Performance' contains three options: 'Performance Hub', 'SQL Tuning Advisor' (which is highlighted with a cursor), and 'SQL Performance Analyzer'. Below this, there's a section titled 'Recommendation Details' with a sub-section 'Stale or Missing Statistics' which provides a brief description about optimizer statistics.

2. Click the **Manual** tab, if needed.
3. Click **MySQLTune** (or the name you gave your tuning task).
4. 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.

This screenshot shows the 'Select Recommendation' section. It lists a single recommendation: 'A potentially better execution plan was found for this statement.' Under the 'Type' column, it is categorized as 'SQL Profile'. To the right, a progress bar indicates a 'Benefit (%)' of 96.9. At the bottom of the table, there are buttons for 'View Details', 'Implement', and 'Validate with SPA'.

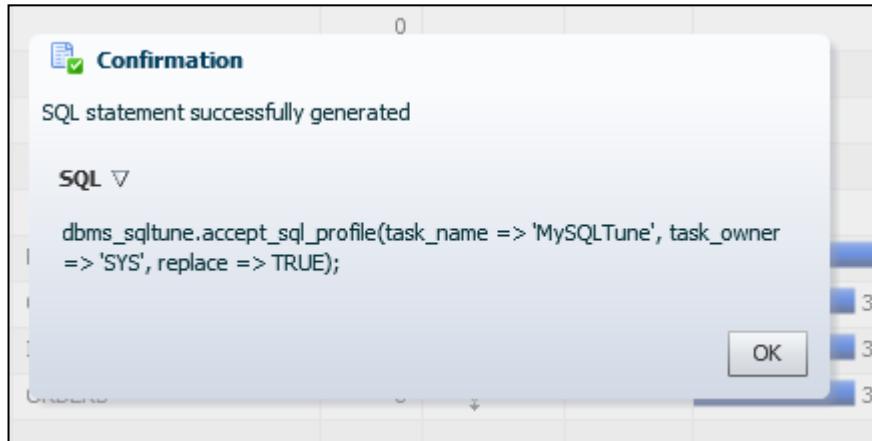
5. In the Select Recommendation section, click the SQL Profile link.
6. On the Original Plan tab, which is displayed by default, view the SQL execution plan. You saw this plan earlier in the practice.
7. Click the Plan Using SQL Profile tab and view its SQL execution plan. Notice the differences between it and the original plan.

This screenshot shows the 'Compare Explain Plans' page. It displays two execution plans: 'Original Plan' and 'Plan Using SQL Profile'. The 'Plan Using SQL Profile' tab is active. The execution plan table includes columns for Operation, Object, Line ID, Predicate, Pruning, Operation Cost, Estimated Rows, and Estimated Bytes. The 'Estimated Bytes' column uses a color scale from dark grey (low) to yellow (high). The 'Estimated Rows' column also uses a color scale. The 'Operation Cost' column shows values like 1, 5, 3, etc.

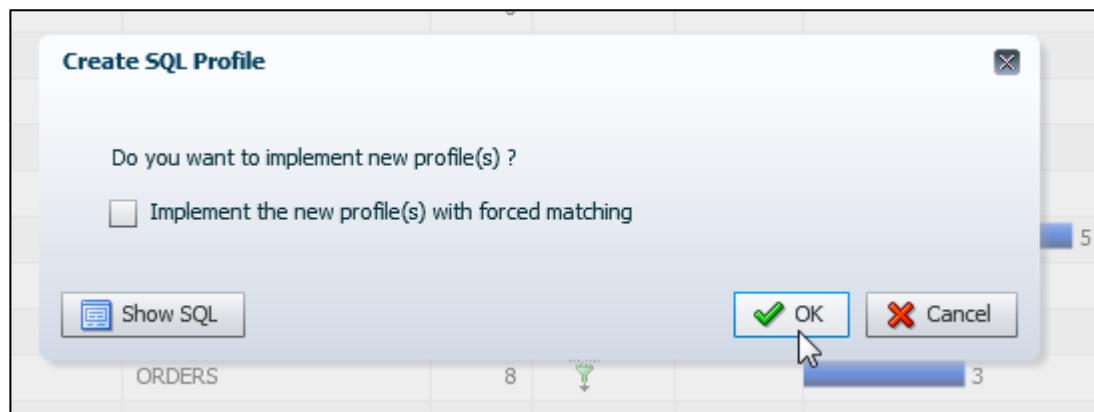
Operation	Object	Line ID	Predicate	Pruning	Operation Cost	Estimated Rows	Estimated Bytes
SELECT STATEMENT		0			1	105	4,515
HASH GROUP BY		1			1	105	4,515
HASH JOIN		2			124	5,332	
HASH JOIN		3			159	3,816	
HASH JOIN		4			99	1,782	
TABLE ACCESS FULL	PRODUCT_INFORMATION	5			86	516	
TABLE ACCESS FULL	ORDER_ITEMS	6			665	7,980	
TABLE ACCESS FULL	INVENTORIES	7			506	3,036	
TABLE ACCESS FULL	ORDERS	8			82	1,558	

8. At the top of the page, click the **Implement** button.
9. In the Create SQL Profile dialog box, click **Show SQL**.

10. A Confirmation dialog box shows you the generated SQL statement. Click **OK**.



11. In the Create SQL Profile dialog box, click **OK** to implement the new profile.



12. 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.

13. Return to the terminal window.
14. If the script is still running, press `Ctrl+c` to stop the activity.
15. Start SQL*Plus and connect to the CDB root as the `SYS` user with the `SYSDBA` privilege.

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

```
...  
SQL>
```

16. 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>
```

17. Remove any blocks of the tables (selected in the query) from the buffer cache.

```
SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;
```

```
System altered.
```

```
SQL>
```

18. Exit SQL*Plus.

```
SQL> EXIT
```

```
...
```

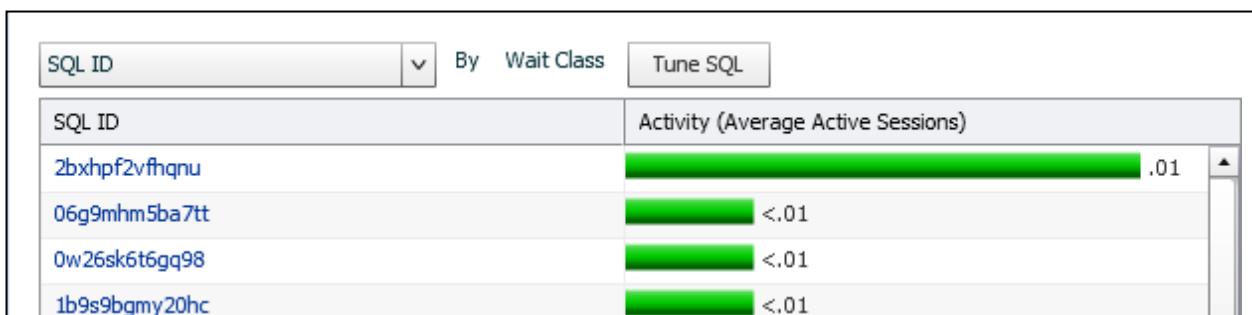
```
[oracle@MYDBCS ~]$
```

19. 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
```

20. Return to EM Express.

21. If a Warning dialog box is displayed stating that a particular SQL ID is no longer from Cursor Cache, click **OK**.
22. Select **Performance** and then **Performance Hub**.
23. Click the **Activity** tab.
24. At the bottom of the page, note the value in the Activity column.



25. 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.

26. Click the link for the SQL ID. The Summary tab is displayed for the SQL ID.
27. Click the **Execution Statistics** tab.

The screenshot shows a browser window with the Oracle Enterprise Manager Express interface. At the top, there are four tabs: 'Summary' (which is selected and highlighted in blue), 'Activity', 'Execution Statistics', and 'Monitored SQL'. Below the tabs, there's a section titled 'Overview' containing the following information:

SQL ID	2bxhpf2vfhqnu
SQL Text	SELECT /*+ monitor USE_NL(d pi i) FULL(d) FULL(i) FULL(pi) */ l.order_id, SUM(uni
First Seen In AWR	Tue Apr 17, 2018 11:01:01 AM
Last Seen In AWR	Tue Apr 17, 2018 11:01:01 AM

28. 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).
29. Click **Log Out** to exit EM Express and close the browser window.
30. 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 ~] $
```

2. 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 ~]$
```

2. 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>
```

3. 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>
```

4. 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>
```

5. Verify that the procedure completed successfully.

- a. Query the USER_ADVISOR_TASKS view.

```

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
  2   FROM user_advisor_tasks
  3 WHERE task_name = 'MY_TASK';

ADVISOR_NAME          EXECUTION_TYPE    LAST_EXECUTION    STATUS
-----  -----  -----  -----
Statistics Advisor      STATISTICS      EXEC_31        COMPLETED
SQL>
```

- b. 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.

```

SQL> SELECT task_name, execution_name, execution_end,
execution_type AS type, status
  2   FROM user_advisor_executions;

TASK_NAME          EXECUTION_ EXECUTION_TYPE
STATUS
-----  -----  -----
-----  -----
AUTO_STATS_ADVISOR_TASK      EXEC_1        07-FEB-18  STATISTICS
COMPLETED
```

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.

- a. 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>

- b. 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>
```

7. 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>
```

8. Generate the script before a possible implementation.

- a. 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>
```

- b. 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>
```

9. Question: What does `obj_filter_list` indicate?

Answer: The object filter list contains the names of the ten objects with no statistics.

10. 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.

11. 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>
```

12. 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.

```
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';
```

TABLE_NAME	NUM_ROWS	BLOCKS	EMPTY_BLOCKS
AVG_ROW_LEN	LAST_ANAL		
ORDERS	105	5	0
38 18-APR-18			
ORDER_ITEMS	665	5	0
18 18-APR-18			

CUSTOMERS	319	13	0
158 18-APR-18			
INVENTORIES	636	5	0
11 18-APR-18			
PRODUCT_INFORMATION	287	13	0
216 18-APR-18			
LINEORDER	3297032	47229	0
98 18-APR-18			
PART	1600000	20645	0
87 18-APR-18			
SUPPLIER	16000	260	0
102 18-APR-18			
CUSTOMER	240000	3845	0
107 18-APR-18			
DATE_DIM	2556	43	0
100 18-APR-18			

10 rows selected.

SQL>

13. Drop the advisor task.

```
SQL> exec DBMS_STATS.DROP_ADVISOR_TASK('my_task')
```

PL/SQL procedure successfully completed.

SQL>

14. 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.

