

Oracle Database 19c: Administration

Table of Contents

Course Practice Environment: Security Credentials	7
Course Practice Environment: Security Credentials.....	8
Practices for Lesson 1: Introduction to Oracle Database	11
Practices for Lesson 1	12
Practices for Lesson 2: Accessing an Oracle Database	13
Practices for Lesson 2	14
Practices for Lesson 3: Creating an Oracle Database by Using DBCA	15
Practices for Lesson 3: Overview	16
Practice 3-1: Creating a New CDB	17
Practices for Lesson 4: Creating an Oracle Database by Using a SQL Command	23
Practices for Lesson 4: Overview	24
Practice 4-1: Creating a New CDB	25
Practices for Lesson 5: Starting Up and Shutting Down an Oracle Database	31
Practices for Lesson 5: Overview	32
Practice 5-1: Shutting Down and Starting Up the Oracle Database	33
Practices for Lesson 6: Managing Database Instances	41
Practices for Lesson 6: Overview	42
Practice 6-1: Investigating Initialization Parameter Files	43
Practice 6-2: Viewing Initialization Parameters by Using SQL*Plus	50
Practice 6-3: Modifying Initialization Parameters by Using SQL*Plus	64
Practice 6-4: Viewing Diagnostic Information	73
Practices for Lesson 7: Oracle Net Services Overview	81
Practices for Lesson 7	82
Practices for Lesson 8: Configuring Naming Methods	83
Practices for Lesson 8: Overview	84
Practice 8-1: Configuring the Oracle Network to Access a Database	85
Practice 8-2: Creating a Net Service Name for a PDB	89
Practices for Lesson 9: Configuring and Administering the Listener	93
Practices for Lesson 9: Overview	94
Practice 9-1: Exploring the Default Listener	95
Practice 9-2: Creating a Second Listener	107
Practice 9-3: Connecting to a Database Service Using the New Listener.....	115
Practices for Lesson 10: Configuring a Shared Server Architecture.....	117
Practices for Lesson 10: Overview	118

Practice 10-1: Configuring Shared Server Mode	119
Practice 10-2: Configuring Clients to Use a Shared Server	122
Practices for Lesson 11: Configuring Oracle Connection Manager for Multiplexing and Access Control	125
Practices for Lesson 11: Overview	126
Practice 11-1: Installing Oracle Instant Client.....	127
Practice 11-2 Configuring Connection Manager.....	130
Practice 11-2: Configuring the Database for Oracle Connection Manager	136
Practice 11-3: Configuring Clients for Oracle Connection Manager.....	138
Practice 11-4: Configuring the Oracle Database Server for Session Multiplexing	140
Practices for Lesson 12: Creating PDBs from Seed	145
Practices for Lesson 12: Overview	146
Practice 12-1: Creating a New PDB from the PDB Seed	147
Practices for Lesson 13: Using Other Techniques to Create PDBs	151
Practices for Lesson 13: Overview	152
Practice 13-1: Cloning Remote PDBs in Hot Mode	153
Practice 13-2: Relocating PDBs.....	160
Practices for Lesson 14: Managing PDBs	169
Practices for Lesson 14: Overview	170
Practice 14-1: Renaming a PDB	171
Practice 14-2: Setting Parameter Values for PDBs	174
Practices for Lesson 15: Database Storage Overview	179
Practices for Lesson 15: Overview	180
Practices for Lesson 16: Creating and Managing Tablespaces	181
Practices for Lesson 16: Overview	182
Practice 16-1: Viewing Tablespace Information	183
Practice 16-2: Creating a Tablespace	192
Practice 16-3: Managing Temporary and Permanent Tablespaces	200
Practices for Lesson 17: Improving Space Usage	207
Practices for Lesson 17: Overview	208
Practice 17-1: Managing Space in Tablespaces	209
Practice 17-2: Using Compression	220
Practice 17-3: Enabling the Resumable Space Allocation Feature	228
Practices for Lesson 18: Managing Undo Data	235
Practices for Lesson 18: Overview	236
Practice 18-1: Managing Undo Tablespaces in a PDB	237
Practices for Lesson 19: Creating and Managing User Accounts	239
Practices for Lesson 19: Overview	240

Practice 19-1: Creating Common and Local Users	241
Practice 19-2: Creating a Local User for an Application	249
Practice 19-3: Exploring OS and Password File Authentication.....	252
Practices for Lesson 20: Configuring Privilege and Role Authorization.....	257
Practices for Lesson 20: Overview	258
Practice 20-1: Granting a Local Role (DBA) to PDBADMIN	259
Practice 20-2: Using SQL*Developer to Create Local Roles.....	262
Practices for Lesson 21: Configuring User Resource Limits	271
Practices for Lesson 21: Overview	272
Practice 21-1: Using SQL*Developer to Create a Local Profile.....	273
Practice 21-2: Using SQL*Developer to Create Local Users	284
Practice 21-3: Configuring a Default Role for a User.....	296
Practices for Lesson 22: Implementing Oracle Database Auditing	301
Practices for Lesson 22: Overview	302
Practice 22-1: Enabling Unified Auditing	303
Practice 22-2: Creating Audit Users.....	307
Practice 22-3: Creating an Audit Policy	309
Practices for Lesson 23: Introduction to Loading and Transporting Data.....	313
Practices for Lesson 23	314
Practices for Lesson 24: Loading Data	315
Practices for Lesson 24: Overview	316
Practice 24-1: Loading Data into a PDB from an External File.....	317
Practices for Lesson 25: Transporting Data	335
Practices for Lesson 25: Overview	336
Practice 25-1: Moving Data from One PDB to Another PDB	337
Practice 25-2: Transporting a Tablespace	352
Practices for Lesson 26: Using External Tables to Load and Transport Data	359
Practices for Lesson 26: Overview	360
Practice 26-1: Querying External Tables	361
Practice 26-2: Unloading External Tables	369
Practices for Lesson 27: Automated Maintenance Tasks Overview.....	373
Practices for Lesson 27	374
Practices for Lesson 28: Managing Tasks and Windows.....	375
Practices for Lesson 28: Overview	376
Practice 28-1: Enabling and Disabling Automated Maintenance Tasks	377
Practice 28-2: Modifying the Duration of a Maintenance Window	379
Practices for Lesson 29: Database Monitoring and Performance Tuning Overview.....	383
Practices for Lesson 29	384

Practices for Lesson 30: Monitoring Database Performance	385
Practices for Lesson 30: Overview	386
Practice 30-1: Using Enterprise Manager Database Express to Manage Performance.....	387
Practices for Lesson 31: Processes.....	399
Practices for Lesson 31: Overview	400
Practice 31-1: Examining the Database Background Processes	401
Practice 31-2: Identifying the Database Server Processes	407
Practices for Lesson 32: Tuning Database Memory.....	413
Practices for Lesson 32: Overview	414
Practice 32-1: Viewing Memory Configurations	415
Practices for Lesson 33: Analyzing SQL and Optimizing Access Paths.....	421
Practices for Lesson 33: Overview	422
Practice 33-1: Using the SQL Tuning Advisor	423
Practice 33-2: Using the Optimizer Statistics Advisor.....	440

Course Practice Environment: SecurityCredentials

Course Practice Environment: Security Credentials

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

Product-Specific Credentials		
Product/Application	Username	Password
Database	SYS	fenago
Database	SYSTEM	fenago
Database	SH, OE, OETEST, BAR	fenago
Database-Practice 3-1 Step 3 all passwords in DBCA command	PDBADMIN	fenago
Database	PDB3_ADMIN	fenago
Database-Practice 13-2 Step 2	TEST	fenago
Database-Practice 14-2 Step 5	ADMIN	fenago
Database/ orclpdb1	PDBADMIN	fenago
Database / newpdb	ADMIN	fenago
Database	C##U	fenago
Database / orclpb1	LU	fenago
Database / orclpb1	HR	fenago
Database	PDBADMIN	fenago
Database	C##CDB_ADMIN1	fenago
Database	PDB1_ADMIN	fenago
Database-Practice 19-2	inventory	fenago
Database-Practice 21-2	JGOODMAN	fenago

Database-Practice 21-2	DHAMBY	Initial password: oracle_4U On password change: fenago
Database-Practice 21-2	RPANDYA	Initial password: oracle_4U On password change: fenago
Database	C##AUDMGR	fenago
Database	C##AUDVWR	fenago
Database	OE	fenago

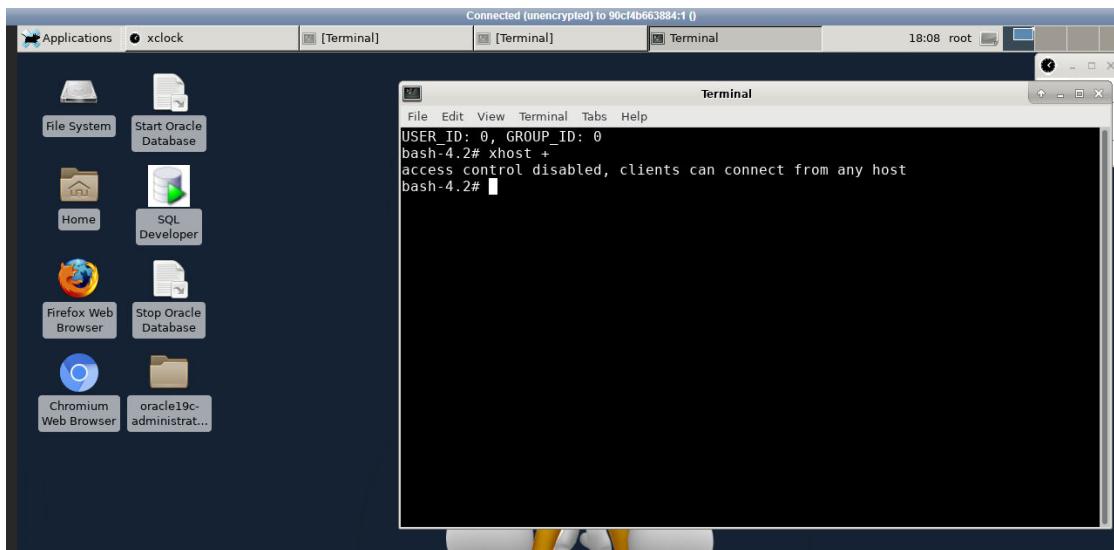
Switch to oracle user from terminal

Overview

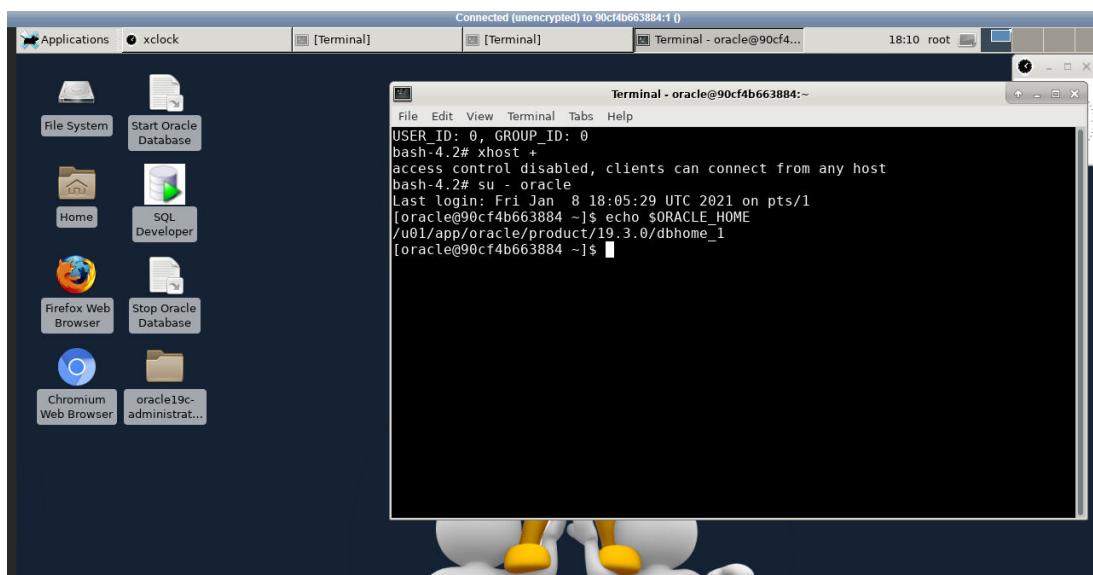
In this practice, you will switch to oracle user from terminal

Tasks

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



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



Note: Open [Install_Database_PDB.pdf](#) and install database/PDB as per instructions before proceeding.

Practices for Lesson 1:Introduction to Oracle Database

Practices for Lesson 1

There are no practices for Lesson 1.

Practices for Lesson 2: Accessing an Oracle Database

Practices for Lesson 2

There are no practices for Lesson 2.

Practices for Lesson 3: Creating an Oracle Database by Using DBCA

Practices for Lesson 3: Overview

Practices Overview

In this practice, you will use Database Configuration Assistant (DBCA) to create a new CDB.

Practice 3-1: Creating a New CDB

Overview

In this practice, you will create a new CDB named `CDBTEST` by using DBCA in silent mode. This CDB will have the following characteristics:

- The users `SYS` and `SYSTEM` will have the same password as the one used for the same users in `orclcdb`. See the *Course Practice Environment: Security Credentials* for the password.
- Oracle Managed Files (OMF) is used for data and redo log files. Set the location of these files to the `/u01/app/oracle/oradata/CDBTEST` directory.
- The CDB root will contain:
 - A default temporary tablespace named `TEMP`
 - A default permanent tablespace named `USERS`
 - An undo tablespace named `UNDOTBS`
- The port used for EM Express is `5502`.
- The CDB is created with no PDBs, except the PDB seed.

Tasks

1. As the oracle OS user, start the existing database instance and listener with `dbstart.sh`.
 - a. Check the status of the database instance.

```
$ pgrep -lf orclcdb
$
```

- b. If nothing is returned, start `orclcdb` database instance and listener. The listener is required for later tasks.

```
$ dbstart.sh
...
$
```

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

```
$ cat /etc/oratab
...
Multiple entries with the same $ORACLE_SID are not allowed.
#
#
orclcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N
rcatcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N
$
```

3. Execute the `$HOME/labs/DBMod_CreateDB/glogin.sh` shell script to set formatting the output.

```
$ $HOME/labs/DBMod_CreateDB/glogin.sh
```

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

4. Create the CDB by using DBCA in silent mode.

- a. Change directory to /home/oracle/labs/DBMod_CreateDB

```
d /home/oracle/labs/DBMod_CreateDB
```

- b. Use the script, /home/oracle/labs/DBMod_CreateDB/CrCDBTEST.sh, as a template; edit it with gedit or your favorite editor. As you edit, change **password** to the password shown in the *Course Practice Environment: Security Credentials*. Save the script. Use the editor you are most comfortable, vi or gedit . The following example with use the notepad like editor gedit

While Linux uses a “-“ as a continuation character, that is not the case in the **CrCDBTEST.sh** script. The dashes are part of the dbca command flags and are not continuation characters. They are part of the command and should be joined with the word on the next line, e.g. -templateName or -useLocalUndoForPDBs

```
$ gedit CrCDBTEST.sh
```

```
...
```

```
$ORACLE_HOME/bin/dbca -silent -createDatabase -  
templateName General_Purpose.dbc -gdbname CDBTEST -sid  
CDBTEST-createAsContainerDatabase true -numberOfPDBs 0 -  
useLocalUndoForPDBs true -responseFile NO_VALUE -totalMemory  
1800 -sysPassword password-systemPassword password -  
pdbAdminPassword password -emConfiguration DBEXPRESS -  
dbsnmpPassword password -emExpressPort 5502 -enableArchive  
true -recoveryAreaDestination  
/u01/app/oracle/fast_recovery_area -recoveryAreaSize 15000 -  
datafileDestination /u01/app/oracle/oradata
```

- c. Verify the edits

```
$ cat CrCDBTEST.sh
```

```
$ORACLE_HOME/bin/dbca -silent -createDatabase -  
templateName General_Purpose.dbc -gdbname CDBTEST -sid  
CDBTEST-createAsContainerDatabase true -numberOfPDBs 0 -  
useLocalUndoForPDBs true -responseFile NO_VALUE -totalMemory  
1800 -sysPassword password-systemPassword password -  
pdbAdminPassword password -emConfiguration DBEXPRESS -  
dbsnmpPassword password -emExpressPort 5502 -enableArchive  
true -recoveryAreaDestination  
/u01/app/oracle/fast_recovery_area -recoveryAreaSize 15000 -  
datafileDestination /u01/app/oracle/oradata
```

```
$
```

- d. Change the permissions on the script to make it executable.

```
$ chmod 755 CrCDBTEST.sh  
$
```

- e. Execute the script. The script may take approximately 10-15 minutes to complete.

```
$ ./CrCDBTEST.sh  
Prepare for db operation  
10% complete  
Copying database files  
40% complete  
Creating and starting Oracle instance  
42% complete  
46% complete  
52% complete  
56% complete  
60% complete  
Completing Database Creation  
66% complete  
70% complete  
Executing Post Configuration Actions  
100% complete  
Database creation complete. For details check the logfiles at:  
/u01/app/oracle/cfgtoollogs/dbca/CDBTEST.  
Database Information:  
Global Database Name:CDBTEST  
System Identifier(SID):CDBTEST  
Look at the log file  
"/u01/app/oracle/cfgtoollogs/dbca/CDBTEST/CDBTEST.log" for  
further details.  
$
```

5. Verify that there is a new entry in /etc/oratab.

```
$ cat /etc/oratab  
...  
# Multiple entries with the same $ORACLE_SID are not allowed.  
#  
#  
orclcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N  
rcatcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N  
CDBTEST:/u01/app/oracle/product/19.3.0/dbhome_1:N  
$
```

6. Verify that the characteristics of the database are correct.

- a. Verify that the database is a CDB.

```
$ . oraenv
ORACLE_SID = [orclcdb] ? CDBTEST
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 DBMod_CreateDB]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Oct 26
18:01:44 2020
Version 19.3.0.0.0

(c) 1982, 2019, Oracle. All rights reserved.

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 -
Production
Version 19.3.0.0.0

SQL> select cdb from v$database;

CDB
---
YES

SQL>
```

- b. Verify that the data files are in the correct directory.

```
SQL> col name format a58
SQL> select name from v$datafile order by 1;

NAME
-----
/u01/app/oracle/oradata/CDBTEST/pdbseed/sysaux01.dbf
/u01/app/oracle/oradata/CDBTEST/pdbseed/system01.dbf
/u01/app/oracle/oradata/CDBTEST/pdbseed/undotbs01.dbf
/u01/app/oracle/oradata/CDBTEST/sysaux01.dbf
/u01/app/oracle/oradata/CDBTEST/system01.dbf
/u01/app/oracle/oradata/CDBTEST/undotbs01.dbf
```

```
/u01/app/oracle/oradata/CDBTEST/users01.dbf

7 rows selected.

SQL>
```

- c. Verify that the tablespaces are created.

```
SQL> col tablespace_name format a15
SQL> col contents format a15
SQL> SELECT tablespace_name, contents FROM dba_tablespaces;

TABLESPACE_NAME CONTENTS
-----
SYSTEM          PERMANENT
SYSAUX          PERMANENT
UNDOTBS1        UNDO
TEMP            TEMPORARY
USERS           PERMANENT

SQL>
```

- d. Verify that the port for EM Express is set correctly.

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

DBMS_XDB_CONFIG.GETHTTPSPORT()
-----
5502

SQL>
```

7. Exit SQL*Plus.

```
SQL> exit
$
```


Practices for Lesson 4: Creating an Oracle Database by Using a SQL Command

Practices for Lesson 4: Overview

Practices Overview

In this practice, you will use the `CREATE DATABASE` SQL command to create a new CDB.

Practice 4-1: Creating a New CDB

Overview

In this practice, you will create a new CDB named `CDBDEV` by using the `CREATE DATABASE` SQL command. This CDB will have the following characteristics:

- The users `SYS` and `SYSTEM` will have the same password as the one used for the same users in `orclcdb`. See *Course Practice Environment: Security Credentials* for the password.
- Oracle Managed Files (OMF) is used for data and redo log files. Set the location of these files to the `/u01/app/oracle/oradata` directory.
- The CDB root will contain:
 - A default temporary tablespace named `TEMP`
 - A default permanent tablespace named `USERS`
 - An undo tablespace named `UNDOTBS`
- The port used for EM Express is `5501`.
- The CDB is created with no PDBs, except the PDB seed.

Tasks

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

```
$ cat /etc/oratab
...
#
orclcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N
rcatcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N
CDBTEST:/u01/app/oracle/product/19.3.0/dbhome_1:N
$
```

2. Create the CDB by using the `CREATE DATABASE` command.

- a. Change the directory to `/home/oracle/labs/DBMod_CreateDB`

```
$ cd /home/oracle/labs/DBMod_CreateDB
$
```

- b. Use the editor of your choice (`vi` and `gedit` are available) to edit the file:

`/home/oracle/labs/DBMod_CreateDB/CrCDBDEV.sql`.

Change `password` to the correct password shown in *Course Practice Environment: Security Credentials* before executing the command. Save the file.

```
$ vi CrCDBDEV.sql
...
CREATE DATABASE cdbdev
USER SYS IDENTIFIED BY password
USER SYSTEM IDENTIFIED BY password
```

```
EXTENT MANAGEMENT LOCAL  
DEFAULT TEMPORARY TABLESPACE temp  
DEFAULT TABLESPACE users  
UNDO TABLESPACE undotbs1  
ENABLE PLUGGABLE DATABASE;
```

- c. Set the oracle environment variables using the oraenv script.

```
$ . oraenv  
ORACLE_SID = [CDBTEST] ? CDBDEV  
ORACLE_HOME = [/home/oracle] ? /u01/app/oracle/product/19.3.0/dbhome_1  
The Oracle base remains unchanged with value /u01/app/oracle  
$
```

- d. Create an initialization parameter file named \$ORACLE_HOME/dbs/initCDBDEV.ora from the sample init.ora file.

```
$ cd $ORACLE_HOME/dbs  
$ cp init.ora initCDBDEV.ora  
$
```

- e. Use the editor of your choice (vi and gedit are available) , add the following initialization parameters to the end of the file

```
$ORACLE_HOME/dbs/initCDBDEV.ora.
```

```
$ vi initCDBDEV.ora  
...  
DB_CREATE_FILE_DEST='/u01/app/oracle/oradata'  
ENABLE_PLUGGABLE_DATABASE=true
```

- f. With the editor still open, change the following initialization parameters to:

```
db_name='CDBDEV'  
audit_file_dest='/u01/app/oracle/admin/CDBDEV/adump'  
db_recovery_file_dest='/u01/app/oracle/fast_recovery_area'  
diagnostic_dest='/u01/app/oracle'  
dispatchers='(PROTOCOL=TCP) (SERVICE=CDBDEVXDB)'  
control_files=('/u01/app/oracle/oradata/ora_control01.ctl','/u01/  
/app/oracle/fast_recovery_area/ora_control02.ctl')  
compatible='19.0.0.0'
```

Note: Change the <ORACLE_BASE> to the actual value wherever <ORACLE_BASE> occurs. Change the COMPATIBLE parameter to the database version you are using.

The COMPATIBLE parameter must be at least 12.0.0.0 to create a container database.

- g. Save the modified file.
h. Verify that the DB_CREATE_FILE_DEST, AUDIT_FILE_DEST, and the DB_RECOVERY_FILE_DEST directories exist. The mkdir -p will create the directory if it does not exist and does not report an error if the directory exists. If the ls command returns anything, the directory exists.

```
$ mkdir -p /u01/app/oracle/admin/CDBDEV/adump
$ ls /u01/app/oracle/fast_recovery_area
CDBTEST  ORCLCDB  RCATCDB

$ ls /u01/app/oracle/oradata
CDBTEST  ORCLCDB  RCATCDB

$
```

- i. Start the database instance in NOMOUNT mode.

```
$ sqlplus / AS SYSDBA

Connected to an idle instance.

SQL> STARTUP NOMOUNT
...
SQL>
```

- j. Execute the script with the CREATE DATABASE command. This command will take approximately two minutes.

```
SQL> @/home/oracle/labs/DBMod_CreateDB/CrCDBDEV.sql

Database created.

SQL>
```

- k. If you receive errors from the CREATE DATABASE command, use the SQL*Plus command SHUTDOWN ABORT, correct the errors, and restart with step 2.i. Check for typos in the initialization file.
- l. Execute catalog and catproc scripts. The catalog.sql script takes ~ 3 minutes. The catproc.sql script takes ~30 minutes

Note: ? is shorthand for \$ORACLE_HOME in SQL*Plus.

```
SQL> @$ORACLE_HOME/rdbms/admin/catalog.sql
...
SQL> @$ORACLE_HOME/rdbms/admin/catproc.sql
...
SQL>
```

- m. Exit from SQL*Plus.

```
SQL> EXIT
...
$
```

3. Add a new entry in the /etc/oratab file with the following command.

```
$ echo "CDBDEV:/u01/app/oracle/product/19.3.0/dbhome_1:N" >>
/etc/oratab
$
```

4. Verify the entry was added to the /etc/oratab file using the cat command:

```
$ cat /etc/oratab  
...  
#  
orclcd:/u01/app/oracle/product/19.3.0/dbhome_1:N  
rcatcd:/u01/app/oracle/product/19.3.0/dbhome_1:N  
CDBTEST:/u01/app/oracle/product/19.3.0/dbhome_1:N  
CDBDEV:/u01/app/oracle/product/19.3.0/dbhome_1:N  
$
```

5. Verify that the characteristics of the database are correct.

- a. Set the environment variables for your new database.

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

- b. Verify that the database is a CDB.

```
$ sqlplus / as sysdba  
...  
Version 19.3.0.0.0  
  
SQL> select cdb from v$database;  
  
CDB  
---  
YES  
  
SQL>
```

- c. Verify that the data files are in the correct location.

```
SQL> set pagesize 100  
SQL> column name format a130  
SQL> select name from v$datafile order by 1;  
  
NAME  
-----  
-----  
/u01/app/oracle/oradata/CDBDEV/B29964A7B1066D26E0536310ED0A031  
2/datafile/o1_mf_s  
ysaux_hsgbrmkx_.dbf  
  
/u01/app/oracle/oradata/CDBDEV/B29964A7B1066D26E0536310ED0A031  
2/datafile/o1_mf_s
```

```

system_hsgbrgd1_.dbf

/u01/app/oracle/oradata/CDBDEV/B29964A7B1066D26E0536310ED0A031
2/datafile/o1_mf_u
sers_hsgbrp5k_.dbf

/u01/app/oracle/oradata/CDBDEV/datafile/o1_mf_sysaux_hsgbrlw5_
.dbf
/u01/app/oracle/oradata/CDBDEV/datafile/o1_mf_system_hsgbrfc8_
.dbf
/u01/app/oracle/oradata/CDBDEV/datafile/o1_mf_undotbs1_hsgbro6
7_.dbf
/u01/app/oracle/oradata/CDBDEV/datafile/o1_mf_users_hsgbroj3_.
dbf

7 rows selected.

SQL>

```

Note: When using Oracle Managed Files (OMF), the file names are harder to read, and the PDB names are not included in the directory path. In this case, the `PDB_SEED` files are shown with the GUID, 888B23FF74E42ED6E0530100007F6226 in the path name.

d. Verify that the specified tablespaces are created for the CDB\$ROOT.

Note: Selecting from the `DBA_TABLESPACES` view shows only the tablespaces associated with the current container, in this case the CDB\$ROOT container. Selecting from the `CDB_TABLESPACES` view would show all the tablespaces for all the open containers.

```

SQL> SELECT tablespace_name, contents FROM dba_tablespaces;

TABLESPACE_NAME          CONTENTS
-----
SYSTEM                  PERMANENT
SYSAUX                  PERMANENT
UNDOTBS1                UNDO
TEMP                   TEMPORARY
USERS                  PERMANENT

SQL>

```

- e. Verify that the EM Express port is correctly set

```
SQL> select dbms_xdb_config.gethttpsport() from dual;  
  
DBMS_XDB_CONFIG.GETHTTPSPORT()  
-----  
0  
  
SQL>
```

Note: Manually creating the database does not set the EM Express port. The EM Express port must be set manually.

- f. Exit SQL*Plus.

```
SQL> exit  
...  
$
```

- g. Close all terminal windows.

Practices for Lesson 5: Starting Up and Shutting Down an Oracle Database

Practices for Lesson 5: Overview

Overview

In these practices, you will learn how to shut down and start up an Oracle Database.

Practice 5-1: 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

The practice assumes that the database and listener are running and may have been started in a previous practice.

The database and listener are NOT automatically started when the VM is started. A script `dbstart.sh` is provided to start the database and listener when needed.

The OS command `pgrep -lf smon` will show any databases that are started, and `pgrep -lf tns` will report any listener processes that are running.

Tasks

1. As the oracle OS user, source the `oraenv` script.

```
$ . oraenv
ORACLE_SID = [CDBDEV] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$
```

2. Create a PDB, `orclpdb3` with the script:

```
/home/oracle/labs/DBMod_CreateDB/setup_pdb3.sh
```

Note: ignore any errors for unable to drop objects.

```
$ ./setup_pdb3.sh
...
SQL> ALTER PLUGGABLE DATABASE ORCLPDB3 CLOSE ;
ALTER PLUGGABLE DATABASE ORCLPDB3 CLOSE
*
ERROR at line 1:
ORA-65011: Pluggable database ORCLPDB3 does not exist.

SQL> DROP PLUGGABLE DATABASE ORCLPDB3 INCLUDING DATAFILES;
DROP PLUGGABLE DATABASE ORCLPDB3 INCLUDING DATAFILES
*
ERROR at line 1:
ORA-65011: Pluggable database ORCLPDB3 does not exist.

SQL> !mkdir /u01/app/oracle/oradata/ORCLCDB/orclpdb3
```

```
mkdir: cannot create directory
\u2018/u01/app/oracle/oradata/ORCLCDB/orclpdb3\u2019: File exists

SQL> CREATE PLUGGABLE DATABASE ORCLPDB3
  2  ADMIN USER admin IDENTIFIED BY fenago  ROLES=(CONNECT) 3
FILE_NAME_CONVERT=('/u01/app/oracle/oradata/ORCLCDB/pdbseed','/u01/
app/oracle/oradata/ORCLCDB/orclpdb3');

Pluggable database created.

SQL> alter PLUGGABLE DATABASE ORCLPDB3 open;

Pluggable database altered.

SQL>
SQL> exit
...
SQL> drop user test cascade;
drop user test cascade
*
ERROR at line 1:
ORA-01918: user 'TEST' does not exist

SQL> create user test identified by fenago;

User created.

SQL> grant dba to test;

Grant succeeded.

SQL> create table test.bigtab (label varchar2(30));

Table created.

SQL> begin
```

```
2  for i in 1..10000 loop
3    insert into test.bigtab values ('DATA FROM test.bigtab');
4    commit;
5  end loop;
6  end;
7 /
```

PL/SQL procedure successfully completed.

```
SQL> EXIT
...
$
```

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 `IMMEDIATE` mode. Normal is the default shutdown mode if no mode is specified. During this mode of shutdown, users sessions are terminated and active transactions are rolled back. 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 IMMEDIATE;
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 2013264224 bytes  
Fixed Size 9136480 bytes  
Variable Size 637534208 bytes  
Database Buffers 1358954496 bytes  
Redo Buffers 7639040 bytes
```

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 ORCLPDB3 is open by querying the OPEN_MODE column in the V\$PDBS view.

```
SQL> COLUMN name FORMAT A10
```

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

CON_ID NAME OPEN_MODE

2 PDB\$SEED READ ONLY
3 ORCLPDB1 READ WRITE
4 ORCLPDB2 READ WRITE
5 ORCLPDB3 MOUNTED

```
SQL>
```

13. Did you expect ORCLPDB3 to be open? By default, PDBs are mounted when a CDB is opened. The STATE of the PDB can be saved in DBA_PDB_SAVED_STATES. This value specifies the STATE the PDB should be in after CDB startup.

```
SQL> COLUMN con_name format a16
```

```
SQL> SELECT con_id, con_name, state FROM DBA_PDB_SAVED_STATES;
```

CON_ID CON_NAME STATE

```
-----  
      3 ORCLPDB1          OPEN  
      4 ORCLPDB2          OPEN  
  
SQL>
```

14. Remove the saved states.

- Set the pluggable databases closed.

```
SQL> alter pluggable database all close;  
  
Pluggable database altered.  
  
SQL>
```

- Set the saved states

```
SQL> alter pluggable database all save state;  
  
Pluggable database altered.  
  
SQL>
```

- View the PDB states.

```
SQL> SELECT con_id, con_name, state FROM DBA_PDB_SAVED_STATES;  
  
no rows selected  
  
SQL>
```

15. Verify the behavior of the save states:

- Shut down the CDB.

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

- Start up the CDB.

```
SQL> STARTUP  
...  
SQL>
```

- View the actual states.

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

CON_ID	NAME	OPEN_MODE
2	PDB\$SEED	READ ONLY
3	ORCLPDB1	MOUNTED
4	ORCLPDB2	MOUNTED
5	ORCLPDB3	MOUNTED

SQL>

16. Reset the PDB save states to open:

- a. Set the pluggable databases open.

```
SQL> alter pluggable database all open;

Pluggable database altered.

SQL>
```

- b. Set the saved state for all PDBs.

```
SQL> alter pluggable database all save state;

Pluggable database altered.

SQL>
```

- c. View the saved states.

```
SQL> select con_id, con_name, state from dba_pdb_saved_states;

CON_ID CON_NAME STATE
----- -----
3 ORCLPDB1 OPEN
4 ORCLPDB2 OPEN
5 ORCLPDB3 OPEN

SQL>
```

17. Shut down a single PDB, ORCLPDB3.

```
SQL> alter pluggable database orclpdb3 close;

Pluggable database altered.

SQL>
```

18. View the status of the PDBs.

```
SQL> show pdbs

  CON_ID CON_NAME        STATE
  -----
    3 ORCLPDB1        OPEN
    4 ORCLPDB2        OPEN
    5 ORCLPDB3        OPEN

SQL>
```

19. Remove ORCLPDB3 and drop the data files associated with ORCLPBD3.

```
SQL> drop pluggable database orclpdb3 including datafiles;

Pluggable database dropped.

SQL>
```

20. Exit SQL*Plus.

```
SAL> exit
...
$
```

21. Close all terminals.

Practices for Lesson 6: Managing Database Instances

Practices for Lesson 6: Overview

Overview

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

Practice 6-1: 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.

The `orclcdb` database instance has been started.

Tasks

1. Use `oraenv` to set the environment variables for the `orclcdb` database.

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

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

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

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

```
SQL> show parameter spfile

NAME                           TYPE        VALUE
-----
spfile                         string
/u01/app/oracle/product/19.3.0/dbhome_1/dbs/spfileorclcdb.ora
SQL>
```

4. 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
$
```

- b. Change to the `$ORACLE_HOME/dbs` directory and use the `ls` command to list the contents of the directory.

```
$ cd $ORACLE_HOME/dbs
```

```
$ ls
hc_CDBDEV.dat    init.ora      orapwCDBTEST           spfileorclcdb.ora
hc_CDBTEST.dat   lkCDBDEV     orapworclcdb          spfileratcldb.ora
hc_orclcdb.dat   lkCDBTEST    orapwrcatcdb
hc_rcatcdb.dat   lkORCLCDB   snapcf_orclcdb.f
initCDBDEV.ora   lkRCATCDB   spfileCDBTEST.ora
$
```

Notice that the SPFILE (spfileorclcdb.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. Then exit from the HOST shell back to SQL*Plus.

```
$ more init.ora
#
# $Header: /rdbms/admin/init.ora /main/25 2015/05/14 15:02:30
kasingha Exp $
#
# (c) 1991, 2015, Oracle and/or its affiliates. All rights reserved.
# NAME
#   init.ora
# FUNCTION
# NOTES
# MODIFIED
#       kasingha 05/12/15 - 21041456 - fix header
#       ysarig    02/01/12 - Renaming flash_recovery_area to
#                           fast_recovery_area
#       ysarig    05/14/09 - Updating compatible to 11.2
#       ysarig    08/13/07 - Fixing the sample for 11g
#       atsukerm  08/06/98 - fix for 8.1.
#       hpioao    06/05/97 - fix for 803
#       glavash   05/12/97 - add oracle_trace_enable comment
#       hpioao    04/22/97 - remove ifile=, events=, etc.
#       alingelb  09/19/94 - remove vms-specific stuff
#       dpawson   07/07/93 - add more comments regarded archive
start
#       maporter  10/29/92 - Add vms_sga_use_gblpagfile=TRUE
#       jloaiza   03/07/92 - change ALPHA to BETA
#       danderso  02/26/92 - change db_block_cache_protect to
#                           _db_block_cache_p
#       ghallmar  02/03/92 - db_directory -> db_domain
#       maporter  01/12/92 - merge changes from branch 1.8.308.1
#       maporter  12/21/91 - bug 76493: Add control_files
parameter
```

```

#      wbridge    12/03/91 - use of %c in archive format is
discouraged
#      ghallmar   12/02/91 - add global_names=true,
db_directory=us.acme.com
#      thayes     11/27/91 - Change default for cache_clone
#      jloaiza    08/13/91 -           merge changes from branch
1.7.100.1
#      jloaiza    07/31/91 -           add debug stuff
#      rlim       04/29/91 -           removal of char_is_varchar2
#      Bridge     03/12/91 - log_allocation no longer exists
#      Wijaya     02/05/91 - remove obsolete parameters
#
#####
#####
# 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'
memory_target=1G
processes = 150
audit_file_dest='<ORACLE_BASE>/admin/orcl/adump'
audit_trail ='db'
db_block_size=8192
db_domain=''
db_recovery_file_dest='<ORACLE_BASE>/fast_recovery_area'

```

```

db_recovery_file_dest_size=2G
diagnostic_dest='<ORACLE_BASE>'
dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
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'

$ exit

```

- d. The `initorclcdb.ora` file does not exist. So create it in SQL*Plus. The '!' character is a shortcut for the `host` command.

```

SQL> create pfile='$ORACLE_HOME/dbs/initorclcdb.ora' from
spfile;

File created
SQL> !
$ cd $ORACLE_HOME/dbs
$ 

```

- e. Now use the `cat` or `more` command to view the text initialization parameter file, `initorclcdb.ora`.

```

$ more initorclcdb.ora
orclcdb._data_transfer_cache_size=0
orclcdb._db_cache_size=1291845632
orclcdb._inmemory_ext_roarea=0
orclcdb._inmemory_ext_rwarea=0
orclcdb._java_pool_size=0
orclcdb._large_pool_size=16777216
orclcdb._oracle_base='/u01/app/oracle'#ORACLE_BASE set from
environment
orclcdb._pga_aggregate_target=671088640
orclcdb._sga_target=2013265920
orclcdb._shared_io_pool_size=100663296
orclcdb._shared_pool_size=570425344
orclcdb._streams_pool_size=16777216
orclcdb._unified_pga_pool_size=0
*.audit_file_dest='/u01/app/oracle/admin/orclcdb/adump'
*.audit_trail='db'
*.compatible='19.0.0'

```

```
*.control_files='/u01/app/oracle/oradata/ORCLCDB/control01.ctl',
'/u01/app/oracle/fast_recovery_area/ORCLCDB/control02.ctl'
*.db_block_size=8192
*.db_name='orclcdb'
*.db_recovery_file_dest='/u01/app/oracle/fast_recovery_area'
*.db_recovery_file_dest_size=14970m
*.diagnostic_dest='/u01/app/oracle'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=orclcdbXDB)'
*.enable_pluggable_database=true
*.local_listener='LISTENER_ORCLCDB'
*.nls_language='AMERICAN'
*.nls_territory='AMERICA'
*.open_cursors=300
*.pga_aggregate_target=640m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=1920m
*.undo_tablespace='UNDOTBS1'
$
```

- f. Return to SQL*Plus.

```
$ exit
SQL>
```

5. 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
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

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

```
SQL> host
$
```

- c. Change to the \$ORACLE_HOME/dbs directory.

```
$ cd $ORACLE_HOME/dbs
$
```

- d. Rename the spfileorclcdb.ora file to spfileorclcdb.ora_original . 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 initORCLCDB.ora file (PFILE) to start the database instance.

```
$ mv spfileorclcdb.ora spfileorclcdb.ora_original  
$
```

- e. Return to SQL*Plus.

```
$ exit  
SQL>
```

- f. Start the database instance by using the STARTUP command.

```
SQL> startup  
...
```

- 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>
```

6. 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.
```

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

```
SQL> host  
$
```

- c. Change to the \$ORACLE_HOME/dbs directory.

```
$ cd $ORACLE_HOME/dbs  
$
```

- d. Rename the orig_spfileorclcdb.ora file to spfileorclcdb.ora.

```
$ mv orig_spfileorclcdb.ora spfileorclcdb.ora  
$
```

- e. Return to SQL*Plus.

```
$ exit  
  
SQL>
```

- f. Start the database instance by using the STARTUP command.

```
SQL> startup  
...
```

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

```
SQL> show parameter spfile  
NAME                                     TYPE        VALUE  
-----  
spfile                               string  
/u01/app/oracle/product/<version_number>/dbhome_1/dbs/spfileorcl  
cdb.ora  
SQL>
```

7. Exit SQL*Plus

```
SQL> exit  
...
```

Practice 6-2: 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 as the `oracle` OS 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. Be sure your environment is using the `orclcdb` database.

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

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

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

3. 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      orclcdb
SQL>
```

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

```
SQL> show parameter db_domain
```

NAME	TYPE	VALUE
db_domain	string	
SQL>		

4. 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	/u01/app/oracle/fast_recovery_area
db_recovery_file_dest_size	big integer	14970M
SQL>		

5. 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.

```
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 1920M
sga_min_size                big integer 0
sga_target                  big integer 1920M
unified_audit_sga_queue_size integer    1048576

SQL>
```

6. 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>
```

7. 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	19.0.0
noncdb_compatible	boolean	FALSE

8. 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	VALUE
control_files	string	/u01/app/oracle/oradata/ORCLCDB/control01.ctl, /u01/app/oracle/fast_recovery_area/ORCLCDB/control02.ctl

```
SQL>
```

9. View the `PROCESSES`, `SESSIONS`, and `TRANSACTIONS` initialization parameters.
- 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	40
log_archive_max_processes	integer	4
processes	integer	300

```
SQL>
```

- 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     0

SQL>
```

View Advanced Parameters

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

10. View the TRANSACTIONS parameter. This is an advanced parameter and seldom needs any adjustment. 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 Y 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>
```

11. 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      200  
  
SQL>
```

12. 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>
```

13. 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>
```

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

```
SQL> show parameter shared_pool_size  
  
NAME          TYPE        VALUE  
-----  
shared_pool_size    big integer 0  
  
SQL>
```

15. 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
<hr/>		
db_block_size	integer	8192
SQL>		

16. 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
<hr/>		
db_cache_size	big integer	0
SQL>		

17. 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
<hr/>		
undo_management	string	AUTO
SQL>		

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

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

- a. View the `MEMORY_TARGET` parameter.

```
SQL> show parameter memory_target
```

NAME	TYPE	VALUE
memory_target	big integer	0

```
SQL>
```

- b. View the `MEMORY_MAX_TARGET` parameter.

```
SQL> show parameter memory_max_target
```

NAME	TYPE	VALUE
memory_max_target	big integer	0

```
SQL>
```

19. 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. 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	640M

```
SQL>
```

Query Views for Parameter Values

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

20. 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 pagesize 100
SQL> select table_name from dictionary where table_name like
'%PARAMETER%';
```

```
TABLE_NAME
```

```

USER_ADVISOR_EXEC_PARAMETERS
USER_ADVISOR_PARAMETERS
...
V$PARAMETER
V$PARAMETER_VALID_VALUES
V$SYSTEM_RESET_PARAMETER2
V$SPPARAMETER
V$SYSTEM_PARAMETER
V$SYSTEM_PARAMETER2
V$SYSTEM_RESET_PARAMETER

66 rows selected.

SQL>

```

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

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

SQL> describe v\$parameter		
Name	Null?	Type
NUM		NUMBER
NAME		VARCHAR2 (80)
TYPE		NUMBER
VALUE		VARCHAR2 (4000)
DISPLAY_VALUE		VARCHAR2 (4000)
DEFAULT_VALUE		VARCHAR2 (255)
ISDEFAULT		VARCHAR2 (9)
ISSES_MODIFIABLE		VARCHAR2 (5)
ISSYS_MODIFIABLE		VARCHAR2 (9)
ISPDB_MODIFIABLE		VARCHAR2 (5)
ISINSTANCE_MODIFIABLE		VARCHAR2 (5)

ISMODIFIED	VARCHAR2 (10)
ISADJUSTED	VARCHAR2 (5)
ISDEPRECATED	VARCHAR2 (5)
ISBASIC	VARCHAR2 (5)
DESCRIPTION	VARCHAR2 (255)
UPDATE_COMMENT	VARCHAR2 (255)
HASH	NUMBER
CON_ID	NUMBER
SQL>	

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

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

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

NAME	ISSYS_MOD	VALUE
DBFIPS_140	FALSE	FALSE
active_instance_count	FALSE	
...		
transactions	FALSE	519
transactions_per_rollback_segment	FALSE	5
...		
wallet_root	FALSE	
workarea_size_policy	IMMEDIATE	AUTO
xml_db_events	IMMEDIATE	enable
445 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 all of the parameters that contain the string "pool."

Note: The values shown may vary from the values displayed in the output.

```
SQL> select name, value from v$parameter
      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             30198988
memoptimize_pool_size                 0
buffer_pool_keep                      0
buffer_pool_recycle                  0
olap_page_pool_size                  0

9 rows selected.

SQL>
```

22. 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 ISSSPECIFIED column.
- View the columns in the V\$SPPARAMETER view by using the DESCRIBE command.

```
SQL> describe v$spparameter

Name                           Null?    Type
-----
FAMILY                         VARCHAR2(80)
SID                            VARCHAR2(80)
NAME                           VARCHAR2(80)
TYPE                           VARCHAR2(11)
VALUE                          VARCHAR2(255)
DISPLAY_VALUE                  VARCHAR2(255)
ISSSPECIFIED                   VARCHAR2(6)
ORDINALV                       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.

SQL> select name, value from v\$spparameter;

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

shrd_dupl_table_refresh_rate	
multishard_query_data_consistency	
multishard_query_partial_results	

449 rows selected.

SQL>

23. Explore the V\$PARAMETER2 view. This view contains information about the initialization parameters that are currently in effect for the session. For parameters with more than one value assigned such as the control_files parameter, each parameter value will be listed as a row in the view's output. 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.

```
SQL> select name, value from v$parameter2;
```

NAME	VALUE
...	
shrd_dupl_table_refresh_rate	60
multishard_query_data_consistency	strong
multishard_query_partial_results	not allowed
450 rows selected.	

```
SQL>
```

24. 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.

```
SQL> select name, value from v$system_parameter;
```

```
NAME
```

```
VALUE
```

```
-----
```

```
common_user_prefix
multishard_query_data_consistency    strong
multishard_query_partial_results     not allowed
```

```
490 rows selected.
```

```
SQL>
```

25. Exit SQL*Plus.

```
SQL> exit
```

```
...
```

Practice 6-3: 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.

The Oracle environment is set to access the `orclcldb` instance.

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  
...
```

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> column NAME FORMAT A18  
SQL> column VALUE Format A20  
SQL> SELECT name, isses_modifiable, issys_modifiable,  
      ispdb_modifiable, value  
      FROM v$parameter  
      WHERE name = 'nls_date_format';  
  
NAME          ISSSES ISSSYS_MOD ISPDB VALUE  
-----  
nls_date_format    TRUE    FALSE     TRUE
```

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.

Note: `NLS_TERRITORY` is set at installation and can be changed with `ALTER SESSION`.

```
SQL> SELECT name, value FROM v$parameter WHERE name =  
'nls_territory';  
  
NAME          VALUE  
-----  
nls_territory    AMERICA
```

4. Connect to `ORCLPDB1`. Run a simple query against the sample data to view an example of the current default date format in use.

- a. Switch to `ORCLPDB1` by using the `ALTER SESSION` command.

```
SQL> ALTER SESSION SET container = ORCLPDB1;  
  
Session altered.
```

- 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  
...  
Higgins              07-JUN-02  
Gietz                07-JUN-02  
  
107 rows selected.
```

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.
```

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

```
SQL> SELECT last_name, hire_date FROM hr.employees;

LAST_NAME          HIRE_DATE
-----
King                jun 17 2003
Kochhar              sep 21 2005
...
Higgins              jun 07 2002
Gietz                jun 07 2002

107 rows selected.
```

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
```

8. Disconnect from ORCLPDB1 to end your session.

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

9. Connect to ORCLPDB1 again as the `SYSTEM` user by using the Easy Connect syntax. See *Course Practice Environment: Security Credentials* for the `SYSTEM` user password. The easy connect syntax is `//<full hostname>:<port number>/<service name>`.
- Get the full hostname.

```
SQL> ! hostname -f
edvmr1p0.us.oracle.com
```

- b. Get the listener port number. The `grep` command reduces the lines to just those using TCP or TCPS protocol. Look for the port using TCP.

```
SQL> ! lsnrctl status | grep tcpl | grep PORT

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com
) (PORT=1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=edvmr1p0.us.oracle.co
m) (PORT=5502)) (Security=(my_wallet_directory=/u01/app/oracle/adm
in/CDBTEST/xdb_wallet)) (Presentation=HTTP) (Session=RAW) )

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=edvmr1p0.us.oracle.co
m) (PORT=5500)) (Security=(my_wallet_directory=/u01/app/oracle/adm
in/orclcdb/xdb_wallet)) (Presentation=HTTP) (Session=RAW) )
```

- c. Connect as a user with `sysdba` privileges.

```
SQL> Connect / as sysdba
...
```

- d. Query the `V$SERVICES` view. Discover the network name.

Note: The network name is the name of the fully qualified database instance and is formed by default from the `instance_name` and `db_domain_name`. In this practice environment, the network name is the same as the database service name.

```
SQL> col name format a20
SQL> col network_name format a20
SQL> SELECT name, network_name FROM V$SERVICES WHERE name =
'orclpdb1';
NAME          NETWORK_NAME
-----
orclpdb1      orclpdb1
```

- e. Using values from steps 9a through 9d, construct and use the Easy Connect string to connect to the `orclpdb1` PDB. Replace `<service_name>` in the `connect` command with the `name` value from the previous step. , change `password` to the password shown in the *Course Practice Environment: Security Credentials*.

Note: The '//' characters preceding the `<full hostname>` are optional.

```
SQL> connect system/password@//<full hostname>:<port
number>/<service name>
Connected.
SQL>
```

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

```
SQL> SELECT last_name, hire_date FROM hr.employees;
```

LAST_NAME	HIRE_DATE
<hr/>	
...	
Higgins	07-JUN-02
Gietz	07-JUN-02
107 rows selected.	

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	

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
...
$ sqlplus / as sysdba
...
```

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> column name format A20
SQL> column value format A20
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 40
```

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.
```

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 three parameters that contain the letters `job`. 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
max_datapump_parallel_per_job	string	50

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.
```

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

```
SQL> STARTUP
```

```
...
```

- 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
max_datapump_parallel_per_job	string	50

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> col name format a30			
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

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`.

- a. What happens if you set `SCOPE=BOTH`?

```
SQL> ALTER SYSTEM SET sec_max_failed_login_attempts = 2
      SCOPE=BOTH;
ALTER SYSTEM SET sec_max_failed_login_attempts = 2 SCOPE=BOTH
*
```

```
ERROR at line 1:  
ORA-02095: specified initialization parameter cannot be modified
```

- b. Now set SCOPE=SPFILE and include the comment.

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

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

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.
```

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

```
SQL> startup  
...
```

- 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

- 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> col name format a30
SQL> col update_comment format a30
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.
```

5. Change the SEC_MAX_FAILED_LOGIN_ATTEMPTS parameter value back to its original value.

```
SQL> ALTER SYSTEM SET sec_max_failed_login_attempts = 3 COMMENT=''
SCOPE=SPFILE;

System altered.
```

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

```
SQL> exit
...
```

Practice 6-4: 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 as the `oracle` user.

The Oracle environment is set to access the `orclcdb` database instance.

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.

```
$ sqlplus / as sysdba  
...
```

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/orclcdb/orclcdb/alert`.
 - The path that corresponds to the `Diag Trace` entry is for the text-only version. This path is `/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace`.

```

SQL> col name format a23
SQL> col value format a55
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/orclcdb/orclcdb
Diag Trace      /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace
Diag Alert       /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert
Diag Incident   /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/incident
Diag Cdmp       /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/cdmp
Health Monitor  /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/hm
Default Trace File /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace/orclcd
b_ora_8778.trc

Active Problem Count  0
Active Incident Count 0
ORACLE_HOME         /u01/app/oracle/product/19.3.0/dbhome_1

12 rows selected.

```

3. Exit SQL*Plus.

```

SQL> EXIT
...

```

View the Alert Log

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

- a. Browse to the `/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert` directory.

```
$ cd /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert
```

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

```
$ ls -l
total 2200
-rw-r----- 1 oracle oinstall 2246772 Oct 16 15:13 log.xml
```

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

events as an alternative to displaying the information on an operator's console.

```
$ cat log.xml
...
<msg time='2020-10-16T15:13:30.854+00:00' org_id='oracle' comp_id='rdbms'
type='UNKNOWN' level='16' host_id='edvmr1p0'
host_addr='10.237.16.202' module='sqlplus@edvmr1p0 (TNS V1-V3)' pid='23327'
con_uid='1' con_id='1' con_name='CDB$ROOT'
<txt>ALTER SYSTEM SET sec_max_failed_login_attempts=3 SCOPE=SPFILE;
</txt>
</msg>
```

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

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

```
$ cd /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace
```

- b. The alert_orclcdb.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.

```
$ ls -l *log
-rw-r----- 1 oracle oinstall 484751 Oct 16 15:13
alert_orclcdb.log
```

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

```
$ tail -500 alert_orclcdb.log
=====
2020-10-26T23:09:24.147134+00:00
db_recovery_file_dest_size of 14970 MB is 0.00% 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.
2020-10-26T23:09:26.351238+00:00
Setting Resource Manager plan
SCEDULER[0x4D52]:DEFAULT_MAINTENANCE_PLAN via scheduler window
```

```
Setting Resource Manager CDB plan DEFAULT_MAINTENANCE_PLAN via parameter
```

- d. Change your directory to the home Directory

```
$ cd
```

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.

```
$ adrci
ADRCI: Release 19.0.0.0.0 - Production on Fri Oct 16 21:27:07 2020
(c) 1982, 2019, 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 will prompt you for which alert log file to display, unless you are in the database's diagnostic directory. Choose the alert log for the orclcdb database:
Note: that the alert log file in the vi editor, by default.

```
adrci> show alert
```

Choose the home from which to view the alert log:

```
1: diag/rdbms/cdbtest/CDBTEST
2: diag/rdbms/orclcdb/orclcdb
3: diag/rdbms/rcatcdb/rcatcdb
4: diag/rdbms/cdbdev/CDBDEV
5: diag/tnslsnr/edvmrlp0/listener
6: diag/clients/user_oracle/host_3132364359_110
Q: to quit
```

```
Please select option: 2
```

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

```
...
Pluggable database ORCLPDB2 opened read write
Starting background process CJQ0
Completed: ALTER DATABASE OPEN
CJQ0 started with pid=48, OS id=23503
2020-10-16 15:11:57.173000 +00:00
```

4. Enter **/Starting ORACLE/** and press return. Press **N** (uppercase) 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.

```
...
2020-10-16 15:11:40.039000 +00:00
Starting ORACLE instance (normal) (OS id: 23224)
*****
Sys-V shared memory will be used for creating SGA
*****
...
...
```

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
2020-10-16 15:11:48.546000 +00:00
...
```

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**.

```
...
Completed: ALTER DATABASE MOUNT
ALTER DATABASE OPEN
...
```

7. Exit the **vi** editor by entering **:q!** and pressing **Enter**.
8. At Alert log list, enter **Q** to leave alert log list.
9. Exit adrci.

```
adrci > exit
```

Log DDL Statements in the DDL Log File

1. Determine if DDL logging is enabled in ORCLPDB1. 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.

```
$ sqlplus / as sysdba  
...
```

- b. Switch to PDB1.

```
SQL> ALTER SESSION SET CONTAINER = ORCLPDB1;  
  
Session altered.
```

- 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     FALSE
```

- d. If DDL logging was not enabled, 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.
```

3. Exit SQL*Plus.

```
SQL> EXIT  
...
```

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

```
$ cd /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/log
```

5. List the contents of the log directory.

```
$ ls
ddl  ddl_orclcdb.log  debug  debug.log  hcs  imedb  test
```

6. View the `ddl_orclcdb.log` file by using the `cat` command.

```
$ cat ddl_orclcdb.log
2020-10-16T21:47:00.225652+00:00
diag_adl:CREATE TABLE TEST (name varchar2(15))
2020-10-16T21:47:05.402413+00:00
diag_adl:drop table test
```

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

```
$ cd ddl
$ ls
log.xml
$ cat log.xml
<msg time='2020-10-16T21:47:00.225+00:00' org_id='oracle'
comp_id='rdbms'
msg_id='kpdbLogDDL:24048:2946163730' type='UNKNOWN'
group='diag_adl'
level='16' host_id='edvmr1p0' host_addr='10.237.16.202'
pid='16214' version='2' con_uid='2991365572'
con_id='3' con_name='ORCLPDB1'>
<txt>CREATE TABLE TEST (name varchar2(15))
</txt>
</msg>
...
```

8. Close the terminal window.

Practices for Lesson 7: Oracle Net Services Overview

Practices for Lesson 7

There are no practices for Lesson 7.

Practices for Lesson 8: Configuring Naming Methods

Practices for Lesson 8: Overview

Overview

In these practices, you will configure network files so that you can access a database on another server. You will also configure access to a PDB.

Practice 8-1: Configuring the Oracle Network to Access a Database

Overview

In this practice, you configure your network environment so that you can connect to another database. Use local naming and create a new network service name called `testorcl` that maps to the other database. Test your network changes by attempting to connect to the other database by using the `testorcl` service name.

Assumptions

The following databases exist & are running: `CDBTEST` and `orclcdb`

Tasks

1. Open a terminal and , use `oraenv` to set your environment to your database sid to `orclcdb` .

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

2. Verify the databases `CDBTEST` and `orclcdb` are in `/etc/oratab`.

```
$ more /etc/oratab
...
orclcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N
rcatcdb:/u01/app/oracle/product/19.3.0/dbhome_1:N
CDBTEST:/u01/app/oracle/product/19.3.0/dbhome_1:N
CDBDEV:/u01/app/oracle/product/19.3.0/dbhome_1

$
```

3. Make a copy of your `tnsnames.ora` file. It is in your database

`$ORACLE_HOME/network/admin` directory.

- a. Change the directory to `$ORACLE_HOME/network/admin` and then list your current working directory.

```
$ cd $ORACLE_HOME/network/admin
$ pwd
$ /u01/app/oracle/product/19.3.0/dbhome_1/network/admin
```

- b. Copy the `tnsnames.ora` file to `tnsnames.old`.

```
$ cp tnsnames.ora tnsnames.old
$
```

- c. Enter `ls -l`, if you want to see the copy and its privileges in your directory.

```
$ ls -l
total 20
-rw-r--r-- 1 oracle oinstall 287 Jun 27 2019 listener.ora
```

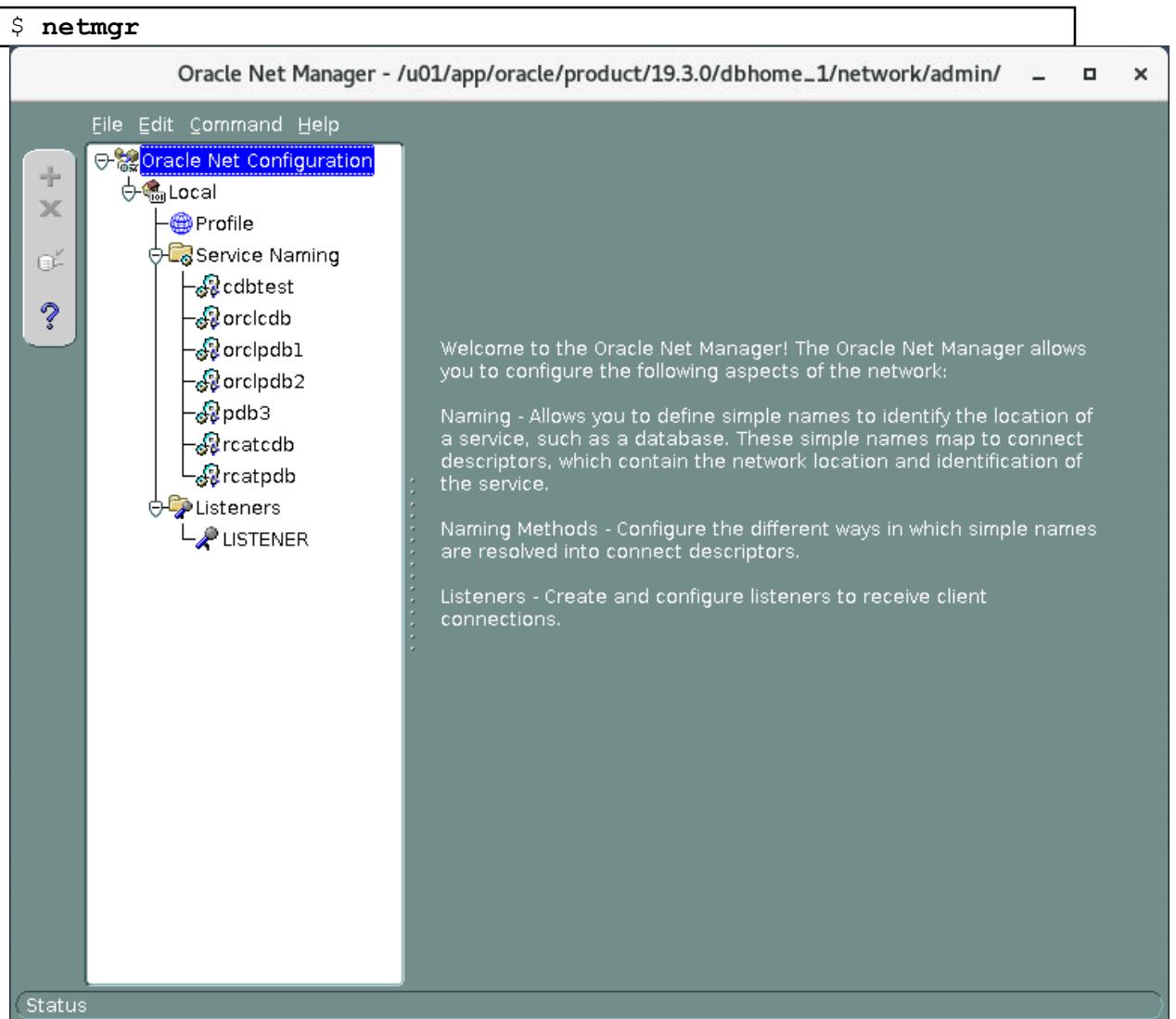
```
drwxr-xr-x 2 oracle oinstall 4096 Apr 17 2019 samples
-rw-r--r-- 1 oracle oinstall 1536 Feb 14 2018 shrept.lst
-rw-r----- 1 oracle oinstall 1870 Oct 16 05:06 tnsnames.ora
-rw-r----- 1 oracle oinstall 1870 Oct 16 22:05 tnsnames.ora_old
$
```

4. Determine the fully qualified host name with the `hostname -f` command. Use the returned value in the following steps.

```
$ hostname -f
edvmrlp0.us.oracle.com
$
```

Use Oracle Net Manager (`netmgr`) to create the `testorcl` net service on your machine.

- Invoke Oracle Net Manager.



- Expand **Local** and select **Service Naming**.
- Click the green **plus sign**.

- d. In the Service Name field, enter `testorcl` and then click **Next**.
 - e. Select **TCP/IP** and then click **Next**.
 - f. In the Host Name field, enter the fully qualified **host name** you found in Step 4.
 - g. In the Port Number field, enter **1521** and then click **Next**.
 - h. In the Service field, enter **CDBTEST**.
 - i. Under Connection type, select **Dedicated Server** and then click **Next**.
 - j. Click **Test**.
 - k. In the “Connection test” dialog box, the test will fail because **SCOTT** is not in the default database. Click **Change Login**.
 - l. In Change Login Box, enter username **system** and **password**. See *Appendix - Product-Specific Credentials* for the password. Click **OK**.
 - m. Click **Test**.
 - n. When "The connection test was successful" message appears, click **Close** and then **Finish**.
 - o. Click **File > Save Network Configuration**.
 - p. Exit Oracle Net Manager.
5. Test your changes to the network configuration by using SQL*Plus. Enter `system@testorcl` and then enter the administrative user password when prompted for the **password**. Select the `INSTANCE_NAME` and `HOST_NAME` columns from the `V$INSTANCE` view to view information about the host.
- a. Ensure your environment is set for the `orclcdb` database by executing the `oraenv` command.
 - b. Invoke SQL*Plus and connect by using the `testorcl` service name.

```
$ sqlplus system@testorcl

Enter password: password

SQL>
```

6. Verify that you are connected to the correct database.

```
SQL> column host_name format a50
SQL> select instance_name, host_name from v$instance;

INSTANCE_NAME      HOST_NAME
-----
CDBTEST            edvmrlp0

SQL>
```

7. Exit SQL*Plus.

```
SQL> exit  
...  
$
```

Practice 8-2: Creating a Net Service Name for a PDB

Overview

In this practice, you create a net service name called MyPDB1 to access a PDB by using Oracle Net Manager.

Tasks

1. Open a terminal window and use oraenv to set the environment variables for the **orclcdb** database.

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

2. Locate and view your local tnsnames.ora file before you add a net service name to it.
 - a. Change the directory to \$ORACLE_HOME/network/admin.

```
$ cd $ORACLE_HOME/network/admin
$
```

- b. List the contents of the current directory. A tnsnames.ora file should be located in this directory.

```
$ ls -l
total 24
-rw-r--r-- 1 oracle oinstall    51 Oct 26 23:49 afiedt.buf
-rw-r--r-- 1 oracle oinstall   301 Oct 23 04:14 listener.ora
drwxr-xr-x 2 oracle oinstall 4096 Apr 17 2019 samples
-rw-r--r-- 1 oracle oinstall 1536 Feb 14 2018 shrept.lst
-rw-r----- 1 oracle oinstall 1908 Oct 26 23:42 tnsnames.old
-rw-r----- 1 oracle oinstall 2186 Oct 26 23:48 tnsnames.ora
$
```

- c. View the tnsnames.ora file by using the cat command. When your CDB and PDB were created, DBCA had automatically created a net service name called orclcdb, which accesses the entire CDB. Later, the PDB1 and PDB2 net service names were added by the developer to make it easy for you to connect to ORCLPDB1 and ORCLPDB2.

```
$ cat tnsnames.ora
...
CDBTEST =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP) (HOST = edvmrlp0) (PORT = 1521))
  )
)
```

```

        (CONNECT_DATA =
          (SERVER = DEDICATED)
          (SERVICE_NAME = CDBTEST)
        )
      )

LISTENER_CDBTEST =
  (ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))
$
```

3. Create a net service name, **MyPDB1**, for **ORCLPDB1** by using Oracle Net Manager.
 - a. Invoke Oracle Net Manager.
 - ```
$ netmgr
```
  - b. Expand **Local** and select **Service Naming**.
  - c. Click the green **plus sign**.
  - d. In the Service Name field, enter **MY1PDB1** and then click **Next**.
  - e. Select **TCP/IP** and then click **Next**.
  - f. In the Host Name field, enter the fully qualified **host name** (hint, step 4 last section) . In the Port Number field, enter **1521** and then click **Next**.
  - g. In the Service field, enter **ORCLPDB1** .
  - h. Under Connection type, select Dedicated Server and then click **Next**.
  - i. Click **Test**.
  - j. In the Connection test dialog box, the test failed because scott does exist. Click **Change Login** and Change Login Box, enter username **system** and **password**. See *Appendix - Product-Specific Credentials* for the password. Click **OK**.
  - k. Click **Test**.
  - l. When "The connection test was successful" message appears, click **Close** and then **Finish**.
  - m. Click **File > Save Network Configuration**.
  - n. Exit Oracle Net Manager.
4. Verify that the entry has been added to the `tnsnames.ora` file.
  - a. Change the directory to `$ORACLE_HOME/network/admin`.
  - ```
$ cd $ORACLE_HOME/network/admin
```
 - ```
$
```

- b. List the contents of the `tnsnames.ora` file by using the `cat` command and verify that the `MYPDB1` net service name entry is listed.

```
$ cat tnsnames.ora
...
MYPDB1 =
 (DESCRIPTION =
 (ADDRESS_LIST =
 (ADDRESS = (PROTOCOL = TCP) (HOST =
edvmr1p0.us.oracle.com) (PORT = 1521))
)
 (CONNECT_DATA =
 (SERVER = DEDICATED)
 (SERVICE_NAME = ORCLPDB1)
)
...
$
```

5. Test the Oracle Net service alias 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 (`PDB1.example.com`) is available.

```
$ tnsping mypdb1

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 16-
OCT-2020 22:52:48
(c) 1997, 2019, Oracle. All rights reserved.

Used parameter files:
Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS =
(PROTOCOL = TCP) (HOST = edvmr1p0.us.oracle.com) (PORT = 1521)))
(CONNECT_DATA = (SERVICE_NAME = ORCLPDB1)))
OK (10 msec)
$
```

6. Connect to ORCLPDB1 and verify the container.

- a. Start SQL\*Plus and connect to `ORCLPDB1` as the system user by using the `MyPDB1` net service name. See *Course Practice Environment: Security Credentials* for the password.

```
$ sqlplus system/password@MyPDB1
...
SQL>
```

- b. Verify that the current container name is ORCLPDB1.

```
SQL> SHOW con_name
CON_NAME

ORCLPDB1
SQL>
```

7. Exit SQL\*Plus.

```
SQL> exit
...
$
```

8. Close the terminal.

# **Practices for Lesson 9: Configuring and Administering the Listener**

## **Practices for Lesson 9: Overview**

---

### **Overview**

In these practices, you will create a new listener and verify that you can connect to a database by using the new listener.

## Practice 9-1: Exploring the Default Listener

---

### Overview

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

### Assumptions

The practice assumes that the database and listener are running and may have been started in a previous practice.

The database and listener are NOT automatically started when the VM is started. A script `dbstart.sh` is provided to start the database and listener when needed.

The OS command `pgrep -lf smon` will show any databases that are started, and `pgrep -lf tns` will report any listener processes that are running.

### Tasks

1. Open a new terminal window and use `oraenv` to set the environment variables for the ORCLCDB database.

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

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

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

3. 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 `orclcdb`, which was named during installation.

```
SQL> SHOW PARAMETER INSTANCE_NAME

NAME TYPE VALUE

instance_name string orclcdb

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, `orclcdb.example.com`, which is a combination of the `DB_NAME` parameter (`orclcdb`) and the `DB_DOMAIN` parameter (`example.com`). If the `DB_DOMAIN` parameter is blank so will the domain portion of the `SERVICE_NAME`. 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                |
|----------------------------|--------|----------------------|
| <code>service_names</code> | string | <code>orclcdb</code> |

```
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. The results show one alias, `LISTENER_ORCLCDB` (`LISTENER_<SID>`). Keep in mind that this isn't the name of the listener. It's an alias for it.

```
SQL> show parameter local_listener
```

| NAME                        | TYPE   | VALUE                         |
|-----------------------------|--------|-------------------------------|
| <code>local_listener</code> | string | <code>LISTENER_ORCLCDB</code> |

```
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 results show that you do not have any remote listeners because the value is null.

```
SQL> show parameter remote_listener
```

| NAME                         | TYPE   | VALUE |
|------------------------------|--------|-------|
| <code>remote_listener</code> | string |       |

```
SQL>
```

4. Exit SQL\*Plus.

```
SQL> exit
...
$
```

5. View the server-side `tnsnames.ora` file and locate the entry that resolves the `LOCAL_LISTENER` parameter value, which is the `LISTENER_ORCLCDB` alias.

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

```
$ cd $ORACLE_HOME/network/admin
$
```

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

```
$ ls
listener.ora samples shrept.lst tnsnames.old tnsnames.ora

$
```

- c. View the `tnsnames.ora` file by using the `less` command (case matters). The entry for the `LISTENER_ORCLCDB` alias contains one protocol address, which consists of a host name, port number (1521, which is the default port number), and protocol (TCP). The protocol address is the listener's "end point." A listener end point does not contain a listener name or a `CONNECT_DATA` section like the `ORCLPDB1` and `ORCLCDB` entries.
- d. **Note:** `less` uses `vi` like key commands to move about the file. End the `less` session with '`q`'.

```
$ less tnsnames.ora
...
tnsnames.ora Network Configuration File:
/u01/app/oracle/product/19.3.0/dbhome_1/network/admin/tnsnames.ora
Generated by Oracle configuration tools.

ORCLCDB =
 (DESCRIPTION =
 (ADDRESS_LIST =
 (ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))
)
 (CONNECT_DATA =
 (SERVER = DEDICATED)
 (SERVICE_NAME = orclcdb)
)
)

LISTENER_ORCLCDB =
 (ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))
```

```

RCATCDB =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))
)
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = rcatcdb)
)
)

LISTENER_RCATCDB =
(ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))

MYPDB1 =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST =
edvmr1p0.us.oracle.com) (PORT = 1521))
)
(CONNECT_DATA =
(SERVICE_NAME = ORCLPDB1)
)
)

ORCLPDB2 =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
)
(CONNECT_DATA =
(SERVICE_NAME = orclpdb2)
)
)

ORCLPDB1 =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521)))
)

```

```

(CONNECT_DATA =
 (SERVICE_NAME = orclpdb1)
)
)

RCATPDB =
(DESCRIPTION =
(ADDRESS_LIST =
 (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
)
(CONNECT_DATA =
 (SERVICE_NAME = rcatpdb)
)
)

PDB3 =
(DESCRIPTION =
(ADDRESS_LIST =
 (ADDRESS = (PROTOCOL = TCP) (HOST = localhost) (PORT =
1521))
)
(CONNECT_DATA =
 (SERVICE_NAME = orclpdb3)
)
)

TESTORCL =
(DESCRIPTION =
(ADDRESS_LIST =
 (ADDRESS = (PROTOCOL = TCP) (HOST =
edvmr1p0.us.oracle.com) (PORT = 1521))
)
(CONNECT_DATA =
 (SERVICE_NAME = cdbtest)
)
)

CDBTEST =
(DESCRIPTION =
(ADDRESS_LIST =
 (ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))
)
)

```

```

 (CONNECT_DATA =
 (SERVER = DEDICATED)
 (SERVICE_NAME = CDBTEST)
)
)

LISTENER_CDBTEST =
 (ADDRESS = (PROTOCOL = TCP) (HOST = edvmrlp0) (PORT = 1521))

$
```

6. View the `listeners.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.

```

$ cat listener.ora
listener.ora Network Configuration File:
/u01/app/oracle/product/19.3.0/dbhome_1/network/admin/listener.ora
Generated by Oracle configuration tools.

LISTENER =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP) (HOST = edvmrlp0) (PORT = 1521))
)

ADR_BASE_LISTENER = /u01/app/oracle

$
```

7. Start the Listener Control utility with the `lsnrctl` command. Without specifying a listener name, the utility assumes you want to connect to the default listener, LISTENER.

```
$ lsnrctl

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 27-OCT-2020
13:56:52

(c) 1991, 2019, Oracle. All rights reserved.Welcome to

LSNRCTL, type "help" for information.

LSNRCTL>
```

8. 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

LSNRCL>
```

- 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.

```
LSNRCTL> status
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0) (PORT=1521)))
STATUS of the LISTENER

Alias LISTENER
Version TNSLSNR for Linux: Version 19.0.0.0.0
- Production
Start Date 15-OCT-2020 17:51:18
Uptime 1 days 5 hr. 17 min. 29 sec
Trace Level off
Security ON: Local OS Authentication
SNMP OFF
Listener Parameter File /u01/app/oracle/product/19.3.0/dbhome_1/network/admin/listener.ora
Listener Log File /u01/app/oracle/diag/tnslsnr/edvmr1p0/listener/alert/log.xml
Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com)
) (PORT=1521))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=edvmr1p0.us.oracle.com)
(PORT=5502)) (Security=(my_wallet_directory=/u01/app/oracle/admin/CDBTEST/xdb_wallet)) (Presentation=HTTP) (Session=RAW))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=edvmr1p0.us.oracle.com)
(PORT=5500)) (Security=(my_wallet_directory=/u01/app/oracle/admin/orclcdb/xdb_wallet)) (Presentation=HTTP) (Session=RAW))
Services Summary...
Service "86b637b62fdf7a65e053f706e80a27ca" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this service...
Service "8857b36632797e5ce0536210ed0adac7" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this service...
Service "8857b419bf707e73e0536210ed0a54c7" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this service...
Service "CDBDEV" has 1 instance(s).
 Instance "CDBDEV", status READY, has 1 handler(s) for this service...
```

```

Service "CDBDEVXDB" has 1 instance(s).
 Instance "CDBDEV", status READY, has 1 handler(s) for this
 service...
Service "CDBTEST" has 1 instance(s).
 Instance "CDBTEST", status READY, has 1 handler(s) for this
 service...
Service "CDBTESTXDB" has 1 instance(s).
 Instance "CDBTEST", status READY, has 1 handler(s) for this
 service...
Service "orclcdb" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
 service...
Service "orclcdbXDB" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
 service...
Service "orclpdb1" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
 service...
Service "orclpdb2" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
 service...
The command completed successfully
LSNRCTL>

```

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

The alias name for the listener is the name that was given to the listener during ORCLCDB creation in DBCA. This alias is entered into the `listeners.ora` file.

If you had named the listener something other than `LISTENER` during CDB creation, the `Alias` value would reflect the other name.

- e. To view additional details about the registered services, issue the `services` command. The results tell you that there are several database services configured for the current listener, three of which are the `orclcdb`, `orclpdb1`, `orclpdb2` services. If the status value for the database instance associated with the database service is `UNKNOWN`, you know that 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 you know that dynamic service registration is going on.

```

LSNRCTL> services
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com
) (PORT=1521)))
Services Summary...
Service "86b637b62fdf7a65e053f706e80a27ca" has 1 instance(s).
 Instance "orclcdb", status READY, has 2 handler(s) for this

```

```

service...

Handler(s):
 "DEDICATED" established:2 refused:0 state:ready
 LOCAL SERVER
 "D000" established:0 refused:0 current:0 max:1022
state:ready
 DISPATCHER <machine: edvmr1p0, pid: 24346>

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=12335)
)

Service "8857b36632797e5ce0536210ed0adac7" has 1 instance(s).
 Instance "orclcdb", status READY, has 2 handler(s) for this
service...
 Handler(s):
 "DEDICATED" established:2 refused:0 state:ready
 LOCAL SERVER
 "D000" established:0 refused:0 current:0 max:1022
state:ready
 DISPATCHER <machine: edvmr1p0, pid: 24346>

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=12335)
)

Service "8857b419bf707e73e0536210ed0a54c7" has 1 instance(s).
 Instance "orclcdb", status READY, has 2 handler(s) for this
service...
 Handler(s):
 "DEDICATED" established:2 refused:0 state:ready
 LOCAL SERVER
 "D000" established:0 refused:0 current:0 max:1022
state:ready
 DISPATCHER <machine: edvmr1p0, pid: 24346>

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=12335)
)

Service "CDBDEV" has 1 instance(s).
 Instance "CDBDEV", status READY, has 1 handler(s) for this
service...
 Handler(s):
 "DEDICATED" established:0 refused:0 state:ready
 LOCAL SERVER
Service "CDBDEVXDB" has 1 instance(s).
 Instance "CDBDEV", status READY, has 1 handler(s) for this
service...
 Handler(s):
 "D000" established:0 refused:0 current:0 max:1022

```

```

state:ready
 DISPATCHER <machine: edvmrlp0, pid: 27937>

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmrlp0.us.oracle.com) (PORT=33359)
)
Service "CDBTEST" has 1 instance(s).
 Instance "CDBTEST", status READY, has 1 handler(s) for this
service...
 Handler(s):
 "DEDICATED" established:4 refused:0 state:ready
 LOCAL SERVER
Service "CDBTESTXDB" has 1 instance(s).
 Instance "CDBTEST", status READY, has 1 handler(s) for this
service...
 Handler(s):
 "D000" established:0 refused:0 current:0 max:1022
state:ready
 DISPATCHER <machine: edvmrlp0, pid: 30421>

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmrlp0.us.oracle.com) (PORT=18685)
)
Service "orclcdb" has 1 instance(s).
 Instance "orclcdb", status READY, has 2 handler(s) for this
service...
 Handler(s):
 "DEDICATED" established:2 refused:0 state:ready
 LOCAL SERVER
 "D000" established:0 refused:0 current:0 max:1022
state:ready
 DISPATCHER <machine: edvmrlp0, pid: 24346>

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmrlp0.us.oracle.com) (PORT=12335)
)
Service "orclcdbXDB" has 1 instance(s).
 Instance "orclcdb", status READY, has 0 handler(s) for this
service...
Service "orclpdb1" has 1 instance(s).
 Instance "orclcdb", status READY, has 2 handler(s) for this
service...
 Handler(s):
 "DEDICATED" established:2 refused:0 state:ready
 LOCAL SERVER
 "D000" established:0 refused:0 current:0 max:1022
state:ready
 DISPATCHER <machine: edvmrlp0, pid: 24346>

```

```

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=12335)
)
Service "orclpdb2" has 1 instance(s).
 Instance "orclcdb", status READY, has 2 handler(s) for this
service...
 Handler(s):
 "DEDICATED" established:2 refused:0 state:ready
 LOCAL SERVER
 "D000" established:0 refused:0 current:0 max:1022
state:ready
 DISPATCHER <machine: edvmr1p0, pid: 24346>

(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=12335)
)
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.

- f. Show the log status. The status is ON, which means the listener activity is being logged.

```

LSNRCTL> show log_status
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com
) (PORT=1521)))
LISTENER parameter "log_status" set to ON
The command completed successfully

LSNRCTL>

```

9. Exit the Listener Control utility and close the terminal window.

```

LSNRCTL> exit
...
$ exit

```

## Practice 9-2: Creating a Second Listener

---

### Overview

In this practice, you create a listener named LISTENER2 that listens on the non-default port 1561 for all database services. Configure the listener to use dynamic service registration, similar to the default listener, LISTENER.

### Assumptions

You are logged in as the `oracle` user.

### Tasks

1. Open the `tnsnames.ora` file and create an entry that resolves a LISTENER2 alias to a protocol address.
  - a. Set your environment variables using oraenv to orclcdb

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

- b. Obtain your host name and domain. The format is `host.domain`

```
$ hostname -f
edvmr1p0.us.oracle.com
$
```

- c. Browse to `$ORACLE_HOME/network/admin`.

```
$ cd $ORACLE_HOME/network/admin
$
```

- d. Copy the `tnsnames.ora` file to `tnsnames.ora.3-2` and then open `tnsnames.ora` in gedit.

```
$ cp tnsnames.ora tnsnames.ora.3-2
$
```

- e. Using the editor of your choice, add an entry to the `tnsnames.ora` file for LISTENER2 to resolve the alias to a protocol address, similar to the `LISTENER_ORCLCDB` entry. You can copy and paste the `LISTENER_ORCLCDB` as a starting point. Specify your host and domain for the host name discovered in step 1b, 1561 for the port number, and TCP as the protocol. The vi editor is shown here.

```
$ vi tnsnames.ora
...
tnsnames.ora Network Configuration File:
/u01/app/oracle/product/12.2.0/dbhome_1/network/admin/tnsnames.ora
Generated by Oracle configuration tools.
```

```

LISTENER2 =
(ADDRESS = (PROTOCOL = TCP)(HOST = edvmr1p0.us.oracle.com) (PORT
= 1561))

LISTENER_ORCLCDB =
(ADDRESS = (PROTOCOL = TCP)(HOST = edvmr1p0.us.oracle.com) (PORT
= 1521))
...

```

- f. Save the file and then exit the editor. Hint: vi write & quit is :wq!
2. Modify the LOCAL\_LISTENER initialization parameter to include both LISTENER\_ORCLCDB and LISTENER2 aliases.
- a. Open a new terminal window and use oraenv to set the environment variables for the orclcdb database.

```

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

```

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

```

$ sqlplus / as sysdba
...
SQL>

```

- c. View the LOCAL\_LISTENER initialization parameter. The value LISTENER\_ORCLCDB is the alias name for the default listener. During dynamic service registration, the LREG process obtains the location of listeners by resolving aliases in the LOCAL\_LISTENER and REMOTE\_LISTENER parameters to entries in the tnsnames.ora file.

```

SQL> SHOW PARAMETER local_listener
NAME TYPE VALUE

local_listener string LISTENER_ORCLCDB
SQL>

```

- d. Check if the `LOCAL_LISTENER` parameter is a static or dynamic parameter by querying the `V$PARAMETER` view. The results tell you that you can't change its value at the session level, but you can at the system level, and the change will take effect immediately. This means that the `LOCAL_LISTENER` parameter is a dynamic system-level parameter.

```
SQL> select isses_modifiable, issys_modifiable from v$parameter
where name='local_listener';

ISSES ISSYS_MOD

FALSE IMMEDIATE

SQL>
```

- e. Set the `LOCAL_LISTENER` parameter equal to `LISTENER_ORCLCDB` and `LISTENER2` by using the `ALTER SYSTEM` command. The change is made to the current instance and is effective immediately.

```
SQL> alter system set
local_listener="LISTENER_ORCLCDB,LISTENER2";

System altered.

SQL>
```

- f. Confirm that `LISTENER_ORCLCDB` and `LISTENER2` are values for the `LOCAL_LISTENER` initialization parameter.

```
SQL> show parameter local_listener

NAME TYPE VALUE

local_listener string LISTENER_ORCLCDB,LISTENER2

SQL>
```

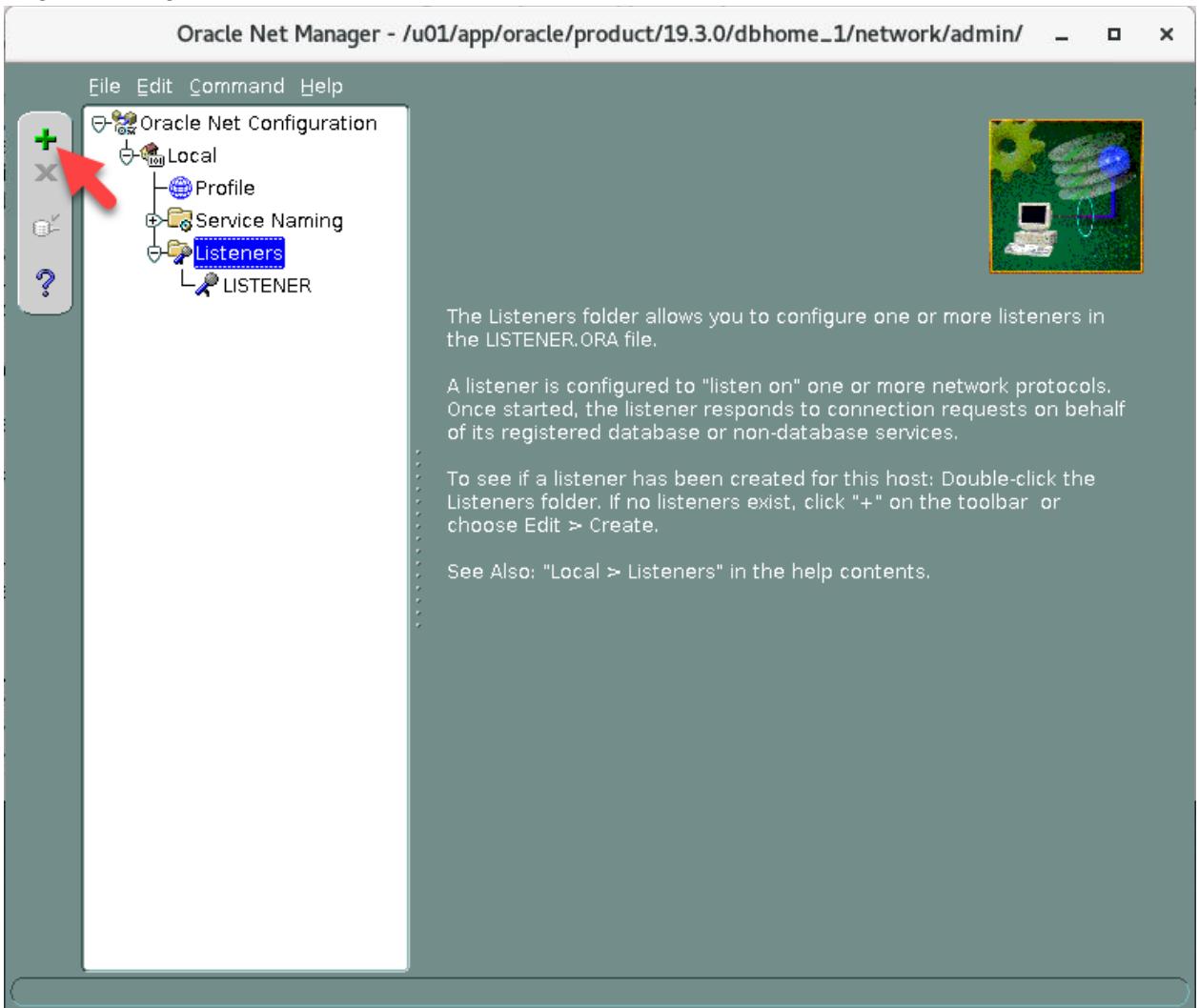
- g. Exit SQL\*Plus. Keep the terminal window open because you will return to it later in the practice.

```
SQL> exit
...
$
```

3. To manage `LISTENER2` with the Listener Control utility, you need to add an entry in the `listener.ora` file so that the utility knows how to connect to `LISTENER2`. In this task, you will use Oracle Net Manager to add the entry. It's important to remember that dynamic service registration does not make use of the `listener.ora` file; however, you do need to configure the file if you want to manage listeners with the Listener Control utility.
- Make a copy of the `listener.ora` file; call it `listener.old`

```
$ cd $ORACLE_HOME/network/admin
$ cp listener.ora listener.old
$
```

- b. In the terminal window, start Oracle Net Manager. Use `netmgr` command.
- c. In the Oracle Net Manager navigation pane, expand **Local** and then **Listeners**. The default listener, LISTENER, is listed. Select **Listeners** and click the green plus sign to begin defining a new listener.



- d. In the Choose Listener Name dialog box, enter **LISTENER2** and click **OK**. LISTENER2 is added to the list of listeners.
- e. With **LISTENER2** selected on the left side, click **Add Address**.
- f. In the Address1 panel on the right, leave Listening Locations selected in the drop-down list, leave TCP/IP selected as the protocol, and set the host name to **your host and domain**. In the Port box, enter **1561**.
- g. For interest, select **General Parameters** from the drop-down list. Review the configuration options on the General, Logging & Tracing, and Authentication tabs.

- h. Select **Database Services** from the drop-down list. There are no databases currently configured for the listener.
  - i. Select **File** and then **Save Network Configuration**.
  - j. Select **File** and then **Exit** to exit Oracle Net Manager. You just added an entry into the `listeners.ora` file.
- k. Change directory (`cd`) to `$ORACLE_HOME/network/admin`**

(`/u01/app/oracle/product/<version>/dbhome_1/network/admin`).

- l. View the `listener.ora` file with the `cat` utility. Notice that the entry for LISTENER2 is added to the file at the top and is configured with the protocol address that you just specified in Oracle Net Manager.**

```
$ cat listener.ora
listener.ora Network Configuration File:
/u01/app/oracle/product/19.3.0/dbhome_1/network/admin/listener.ora
Generated by Oracle configuration tools.

LISTENER2 =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0.us.oracle.com) (PORT = 1561))
)

ADR_BASE_LISTENER2 = /u01/app/oracle

LISTENER =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))
)

ADR_BASE_LISTENER = /u01/app/oracle
$
```

4. Using the Listener Control utility, check the status of LISTENER2.
- a. In the terminal window, start the Listener Control utility.

```
$ lsnrctl
...
LSNRCTL>
```

- b. Check the status of LISTENER2 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 LISTENER2
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com
)(PORT=1561)))
TNS-12541: TNS:no listener
TNS-12560: TNS:protocol adapter error
TNS-00511: No listener
Linux Error: 111: Connection refused

LSNRCTL>
```

- c. Start LISTENER2 by issuing the start LISTENER2 command. The status indicates that the listener does not support any services.

```
LSNRCTL> start listener2
Starting /u01/app/oracle/product/19.3.0/dbhome_1/bin/tnslsnr:
please wait...

TNSLSNR for Linux: Version 19.0.0.0.0 - Production
System parameter file is
/u01/app/oracle/product/19.3.0/dbhome_1/network/admin/listener.o
ra
Log messages written to
/u01/app/oracle/diag/tnslsnr/edvmr1p0/listener2/alert/log.xml
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com
)(PORT=1561)))

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com
)(PORT=1561)))
STATUS of the LISTENER

Alias listener2
Version TNSLSNR for Linux: Version 19.0.0.0.0
- Production
Start Date 16-OCT-2020 23:27:54
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/19.3.0/dbhome_1/network/admin/listener.ora
Listener Log File
/u01/app/oracle/diag/tnslnsr/edvmr1p0/listener2/alert/log.xml
Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com)
) (PORT=1561)))
The listener supports no services
The command completed successfully

LSNRCTL>

```

- d. Question: Why do you think the listener is not supporting any services?

Answer: One reason might be that the LREG process hasn't had enough time to dynamically update the list of services for the listener yet.

- e. Wait about 60 seconds and then check the status of LISTENER2 again. By then, the LREG process will have time to register the database services with your listener.

```

LSNRCTL> status listener2
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com)
) (PORT=1561)))
STATUS of the LISTENER

Alias listener2
Version TNSLSNR for Linux: Version 19.0.0.0.0
- Production
Start Date 16-OCT-2020 23:27:54
Uptime 0 days 0 hr. 0 min. 55 sec
Trace Level off
Security ON: Local OS Authentication
SNMP OFF
Listener Parameter File
/u01/app/oracle/product/19.3.0/dbhome_1/network/admin/listener.ora
Listener Log File
/u01/app/oracle/diag/tnslnsr/edvmr1p0/listener2/alert/log.xml
Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com)
) (PORT=1561)))
Services Summary...
Service "86b637b62fdf7a65e053f706e80a27ca" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
service...

```

```
Service "8857b36632797e5ce0536210ed0adac7" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
service...
Service "8857b419bf707e73e0536210ed0a54c7" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
service...
Service "orclcdb" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
service...
Service "orclcdbXDB" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
service...
Service "orclpdb1" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
service...
Service "orclpdb2" has 1 instance(s).
 Instance "orclcdb", status READY, has 1 handler(s) for this
service...
The command completed successfully

LSNRCTL>
```

- f. Exit the Listener Control utility.

```
LSNRCTL> exit
...
$
```

## Practice 9-3: Connecting to a Database Service Using the New Listener

---

### Overview

Now that you have LISTENER2 configured, test it by making a connection to one of its supported database services, for example, orclcdb.

### Tasks

1. Using Easy Connect syntax, start SQL\*Plus and connect to the CDB using LISTENER2. Make sure to specify the non-default port number 1561. See *Course Practice Environment: Security Credentials* for the password.  
**Note:** The service name would usually include the domain name as specified in the value of the db\_domain initialization parameter. The domain value is blank in your deployment.

```
$ sqlplus system/password@localhost:1561/orclcdb
...
SQL>
```

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

```
SQL> exit
...
$ exit
```



# **Practices for Lesson 10: Configuring a Shared Server Architecture**

## **Practices for Lesson 10: Overview**

---

### **Overview**

In these practices, you will configure shared server mode. Then you will configure a network service to connect to the database using a shared server.

## Practice 10-1: Configuring Shared Server Mode

---

### Overview

In this practice, you configure shared server mode.

### Tasks

1. Open a terminal window and use oraenv to set the environment variables for the `orclcdb` database.

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

2. Log in to SQL\*Plus as a user with `SYSDBA` privileges.

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

3. Determine whether the shared server architecture is implemented in your database.

- a. Check the value of the `SHARED_SERVER` initialization parameter

```
SQL> show parameter shared_server

NAME TYPE VALUE

max_shared_servers integer
shared_server_sessions integer
shared_servers integer 1
SQL>
```

- b. Check the value of the `DISPATCHERS` initialization parameter.

```
SQL> show parameter dispatchers

NAME TYPE VALUE

dispatchers string (PROTOCOL=TCP) (SERVICE=orclcdbXDB)
max_dispatchers integer
SQL>
```

- c. Question: Why is there a shared server and dispatcher?

Answer: If you create your Oracle database with Database Configuration Assistant (DBCA), DBCA configures a dispatcher for Oracle XML DB (XDB). This is because XDB protocols like HTTP and FTP require shared server. This results in a `SHARED_SERVER` value of 1. Although shared server is enabled, this configuration

permits only sessions that connect to the XDB service to use shared server. To enable shared server for regular database sessions (for submitting SQL statements), you must add an additional dispatcher configuration or replace the existing configuration with one that is not specific to XDB.

4. Enable three shared servers in your database.

```
SQL> alter system set shared_servers = 3;

System altered.

SQL>
```

5. Determine whether you need to configure any additional dispatchers to support TCP connections.
  - a. Check the value of the DISPATCHERS initialization parameter.

```
SQL> show parameter dispatchers

NAME TYPE VALUE

dispatchers string (PROTOCOL=TCP) (SERVICE=orclcdbXDB)
max_dispatchers integer

SQL>
```

Question: Do you need to configure any additional dispatchers?

Answer: Yes. When shared server mode is enabled, a dispatcher is started automatically on the TCP/IP protocol even if the DISPATCHERS parameter has not been set. But a dispatcher with a specified service will connect only to that service.

6. Change the dispatcher service so it can connect to any service using TCP/IP.

```
SQL> ALTER SYSTEM SET dispatchers = "(PROTOCOL=TCP)";

System altered.
```

7. Confirm change:

```
SQL> show parameter dispatchers

NAME TYPE VALUE

dispatchers string (PROTOCOL=TCP)
max_dispatchers integer

SQL>
```

8. Exit SQL\*Plus.

```
SQL> exit
...
$
```

9. Close all terminals.

## Practice 10-2: Configuring Clients to Use a Shared Server

---

### Overview

In this practice, you configure a network service that uses shared server.

### Tasks

1. Open a terminal window and use oraenv to set the environment variables for the `orclcdb` database.

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

2. Configure a network service that uses a shared server connection.
  - a. Change the directory to `$ORACLE_HOME/network/admin` and then list your current working directory.

```
$ cd $ORACLE_HOME/network/admin
[oracle@edvmr1p0 admin]$ pwd
/u01/app/oracle/product/19.3.0/dbhome_1/network/admin
$
```

- b. Make a copy of `tnsnames.ora`.

```
$ cp tnsnames.ora tnsnames.ora.4-2
$
```

- c. Use an editor such as `vi` or `gedit` to edit the `tnsnames.ora` file, this example uses `vi`. Add a new service named `test_ss` that uses a dispatcher.

**Hint:** Copy the `ORCLCDB` entry and change the necessary fields

```
$ vi tnsnames.ora
...
TEST_SS =
 (DESCRIPTION=
 (ADDRESS_LIST =
 (ADDRESS=(PROTOCOL=tcp) (HOST=
edvmr1p0.us.oracle.com) (PORT=1521))
)
 (CONNECT_DATA=
 (SERVICE_NAME=ORCLCDB)
 (SERVER=shared)
)
)
```

3. Invoke SQL\*Plus and connect to the database using a dispatcher. See *Course Practice Environment: Security Credentials* for passwords.

Note: call this **Terminal 1**

```
$ sqlplus system@test_ss
```

```
Enter password: password
```

```
...
SQL>
```

4. Open another terminal window and use oraenv to set the environment variables for the orclcdb database. Call this **Terminal 2**

```
$. oraenv
```

```
ORACLE_SID = [orclcdb] ? orclcdb
```

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

5. In **Terminal 2**, log in to SQL\*Plus as a user with SYSDBA privileges.

```
$ sqlplus / as sysdba
```

```
...
SQL>
```

6. In **Terminal 2**, view information about all shared server connections by querying V\$CIRCUIT.

```
SQL> select dispatcher, server, saddr, queue from v$circuit;
```

| DISPATCHER | SERVER | SADDR | QUEUE |
|------------|--------|-------|-------|
|------------|--------|-------|-------|

|                  |    |                 |      |
|------------------|----|-----------------|------|
| 000000009F54D420 | 00 | 00000009F967A18 | NONE |
|------------------|----|-----------------|------|

```
SQL>
```

7. In **Terminal 1**, log out of the SQL\*Plus session that uses the test\_ss service.

8. In **Terminal 2**, view information about the shared server connection by querying V\$CIRCUIT.

```
SQL> select dispatcher, server, saddr, queue from v$circuit;
```

```
no rows selected
```

```
SQL>
```

9. Log out of SQL\*Plus and close both terminal window

---



## **Practices for Lesson 11: Overview**

---

There are no practices for Lesson 11.

# **Practices for Lesson 12: Creating PDBs from Seed**

## **Practices for Lesson 12: Overview**

---

### **Practices Overview**

In this practice, you create a new PDB by using the PDB seed.

## Practice 12-1: Creating a New PDB from the PDB Seed

---

### Overview

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

The new PDB is named `ORCLPDB3`. The PDB should have the following characteristics:

- The users `SYS` and `SYSTEM` will have the same password as the one used for the same users in `orclcdb`. See *Course Practice Environment: Security Credentials* for passwords.
- The DBA user for the PDB is `pdb3_admin`. The `pdb3_admin` user will have the same password as the one used for `SYS` and `SYSTEM`.
- Set the location of the PDB datafiles to the `/u01/app/oracle/oradata/` directory.

### Assumptions:

- The `orclcdb` database instance and listener have been started.

### Tasks

1. Open a terminal and use `oraenv` to set the environment variables for the `orclcdb` database.

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

2. Run the script `/home/oracle/labs/DBMod_PDBs/setup_tns.sh`

```
$ /home/oracle/labs/DBMod_PDBs/setup_tns.sh
The Oracle base remains unchanged with value /u01/app/oracle
$
```

3. As the `oracle` OS user, create the new PDB.

- a. Log in to SQL\*Plus and connect to the CDB root with a user with the `CREATE PLUGGABLE DATABASE` privilege.

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

- b. Execute the `CREATE PLUGGABLE DATABASE` command. Be sure to replace `password` with the correct password. See *Course Practice Environment: Security Credentials* for passwords.

```
SQL> CREATE PLUGGABLE DATABASE orclpdb3
ADMIN USER pdb3_admin IDENTIFIED BY password
ROLES=(CONNECT)
CREATE_FILE_DEST='/u01/app/oracle/oradata' ;
```

```
Pluggable database created.
```

```
SQL>
```

4. Check the status of the new PDB. The `PDB_ID` value may vary from what is shown.

```
SQL> column pdb_name format a16
SQL> select pdb_id, pdb_name, status from cdb_pdbs;
```

| PDB_ID | PDB_NAME  | STATUS |
|--------|-----------|--------|
| 3      | ORCLPDB1  | NORMAL |
| 2      | PDB\$SEED | NORMAL |
| 4      | ORCLPDB2  | NORMAL |
| 5      | ORCLPDB3  | NEW    |

```
SQL>
```

5. Open the new PDB and check the status again. Exit from SQL\*Plus.

```
SQL> alter pluggable database orclpdb3 open;
Pluggable database altered.

SQL> select pdb_id, pdb_name, status from cdb_pdbs;

PDB_ID PDB_NAME STATUS

3 ORCLPDB1 NORMAL
2 PDB$SEED NORMAL
4 ORCLPDB2 NORMAL
5 ORCLPDB3 NORMAL

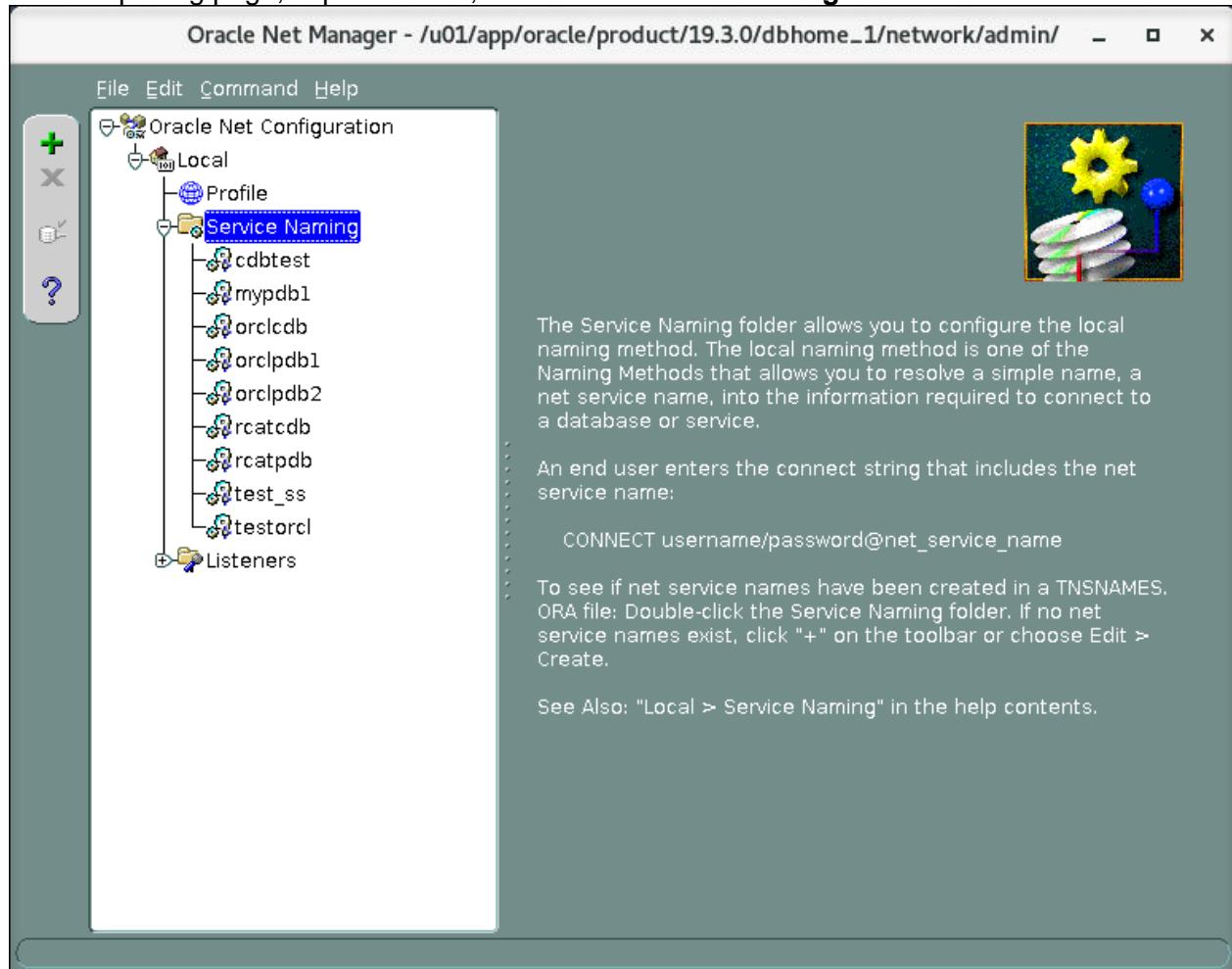
SQL> exit
```

6. Create a net service name, PDB3, for the new PDB.

- a. Launch `netmgr`.

```
$ netmgr
```

- b. On the opening page, expand **Local**, and select **Service Naming**.



- c. Click the **plus sign**.
- d. Enter Net Service Name: **PDB3**, and click **Next**.
- e. Select **TCP/IP (Internet Protocol)**, and click **Next**.
- f. Enter Host Name: **localhost**, verify the Port Number is **1521**, and click **Next**.
- g. Enter Service Name: **ORCLPDB3**
- h. For Connection Type, select **Dedicated Server** and click **Next**.
- i. Click **Finish**.
- j. Click **File>Save Network Configuration**.
- k. Click **File>Exit**.

7. Connect to the PDB and verify that the datafiles are in the correct location. See *Course Practice Environment: Security Credentials* for passwords. The file names will vary as OMF assigns unique filenames.

**Note:** Because OMF is selected by using the `CREATE_FILE_DEST` parameter, the `create` command will use the specified directory appended with the `/database` name/`PDB_ID`.

```
$ sqlplus system@PDB3
...
Enter password: password
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 -
Production
Version 19.3.0.0.0
SQL>
SQL> select name from v$datafile;

NAME

/u01/app/oracle/oradata/ORCLCDB/8A6FC54A03E11CBDE0536210ED0AFC84
/datafile/o1_mf_system_ghbpjdrv_.dbf

/u01/app/oracle/oradata/ORCLCDB/8A6FC54A03E11CBDE0536210ED0AFC84
/datafile/o1_mf_sysaux_ghbpjds8_.dbf

/u01/app/oracle/oradata/ORCLCDB/8A6FC54A03E11CBDE0536210ED0AFC84
/datafile/o1_mf_undotbs1_ghbpjds9_.dbf

SQL>
```

8. Verify that the service is ORCLPDB3. Then exit from SQL\*Plus.

```
SQL> col name format a15
SQL> select name from v$services;

NAME

orclpdb3

SQL> exit
...
$
```

9. Exit the terminal.

# **Practices for Lesson 13: Using Other Techniques to Create PDBs**

## **Practices for Lesson 13: Overview**

---

### **Practices Overview**

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

- Cloning a regular PDB in hot mode and with automatic refreshing
- Relocating a PDB

## Practice 13-1: Cloning Remote PDBs in Hot Mode

---

### Overview

In this practice, because you have been informed of performance issues on the `PDB_SOURCE_FOR_HOTCLONE` PDB in `ORCL`, you will clone the PDB in hot mode as the `pdb_HOTCLONE` PDB in the `CDBTEST` test instance for performance tests. The remote `PDB_SOURCE_FOR_HOTCLONE` production PDB in `ORCLCDB` will still be up and fully functional while the actual clone operation is taking place. At the end of the practice, you will create a refreshable copy, refreshed manually or automatically, which will allow you to take your time to test the performance issue.

### Assumptions:

- CDB `orclcdb` contain at least two pdbs: `orclpdb1`, `orclpdb2` and `orclpdb3`

### Tasks

1. Execute the `/home/oracle/labs/admin/cleanup_PDBs.sh` shell script to prepare your CDB and PDBs for this practice. The shell script drops all PDBs that may have been created in `ORCLCDB`. If the PDBs do not exist, error messages will show that they do not exist.

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

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

```
$ $HOME/labs/DBMod_PDBs/glogin_4.sh
The Oracle base remains unchanged with value /u01/app/oracle
$
```

3. Execute the `$HOME/labs/DBMod_PDBs/setup_hotclone.sh` script which creates the production `PDB_SOURCE_FOR_HOTCLONE` PDB from the PDB seed in `ORCLCDB`, creates `CDBTEST`, creates a local `SOURCE_USER` user in `PDB_SOURCE_FOR_HOTCLONE`, and creates the `SOURCE_USER.BIGTAB` table with thousands of rows. This script will take some time to complete.

```
$ $HOME/labs/DBMod_PDBs/setup_hotclone.sh
The Oracle base remains unchanged with value /u01/app/oracle
Input the SYS user password: password
SQL*Plus: Release 19.0.0.0.0 - Production on Tue Oct 20 16:22:38
2020
...
$
```

4. In your terminal window, set the environment variables to `orclcdb`, then grant privileges to the user who will perform the hot clone operation in this case `SYSTEM`.

As a sysdba privileged user, grant privileges to the system user who will create the hot clone.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL> grant create pluggable database to SYSTEM container = all;
Grant succeeded

SQL> exit
...
$
```

5. In your terminal, log into sqlplus and start a transaction in the PDB\_SOURCE\_FOR\_HOTCLONE PDB. See *Course Practice Environment: Security Credentials* for passwords. Name the terminal window Session ORCLPDB. **Hint:** In the terminal menu, click Terminal > Set Title.

```
$ sqlplus system@pdb_source_for_hotclone
...
Enter password: password

SQL> set sqlprompt "ORCLCDB> "
ORCLCDB> select distinct label from source_user.bigtab;

LABEL

DATA FROM source_user.bigtab

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

10000 rows updated.

SQL>
```

6. In CDBTEST, clone PDB\_HOTCLONE from PDB\_SOURCE\_FOR\_HOTCLONE in hot mode.
  - a. Start a new terminal window, and name it Session CDBTEST. Create the directory needed for PDB\_HOTCLONE.

```
$ mkdir -p /u01/app/oracle/oradata/CDBTEST/pdb_hotclone
$
```

- b. In this terminal window, named Session CDBTEST, Set the sqlprompt to CDBTEST.

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

```
...
SQL> SET sqlprompt "CDBTEST> "
CDBTEST>
```

- c. Clone the PDB\_HOTCLONE PDB from PDB\_SOURCE\_FOR\_HOTCLONE while the source PDB is still up and fully functional. See *Course Practice Environment: Security Credentials* for passwords.

```
CDBTEST> DROP PUBLIC DATABASE LINK link_pdb_source_for_hotclone;
DROP PUBLIC DATABASE LINK link_pdb_source_for_hotclone
*
ERROR at line 1:
ORA-02024: database link not found

CDBTEST> CREATE PUBLIC DATABASE LINK
 link_pdb_source_for_hotclone
 CONNECT TO system IDENTIFIED BY password
 USING 'pdb_source_for_hotclone';

Database link created.

CDBTEST> ALTER SESSION SET db_create_file_dest=
 '/u01/app/oracle/oradata/CDBTEST/pdb_hotclone';

Session altered.

CDBTEST> CREATE PLUGGABLE DATABASE pdb_hotclone
 FROM pdb_source_for_hotclone@link_pdb_source_for_hotclone
 FILE_NAME_CONVERT=
 ('/u01/app/oracle/oradata/ORCLCDB/pdb_source_for_hotclone',
 '/u01/app/oracle/oradata/CDBTEST/pdb_hotclone')
 REFRESH MODE MANUAL;

Pluggable database created.

CDBTEST>
```

7. Open PDB\_HOTCLONE in READ ONLY mode only.

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

Pluggable database altered.

CDBTEST>
```

8. Select the same data from SOURCE\_USER.BIGTAB in the cloned PDB.

```
CDBTEST> ALTER SESSION SET container = pdb_hotclone;

Session altered.

CDBTEST> select distinct label from source_user.bigtab;

LABEL

DATA FROM source_user.bigtab

CDBTEST> select count(*) from source_user.bigtab;

COUNT (*)

10000

CDBTEST>
```

9. In Session ORCLPDB, commit the updated and confirm the change.

```
ORCLCDB> commit;

Commit complete.

ORCLCDB> select distinct label from source_user.bigtab;

LABEL

DATA

ORCLCDB>
```

10. In Session CDBTEST, refresh the data in the cloned PDB in CDBTEST.

```
CDBTEST> alter pluggable database pdb_hotclone refresh;
ALTER PLUGGABLE DATABASE pdb_hotclone REFRESH
*
ERROR at line 1:
ORA-65025: Pluggable database PDB_HOTCLONE is not closed on all
instances.

CDBTEST>
```

**Note:** The refreshable copy PDB must be closed in order for refresh to be performed. If it is not closed when automatic refresh is attempted, the refresh will be deferred until the next scheduled refresh.

```
CDBTEST> ALTER PLUGGABLE DATABASE pdb_hotclone CLOSE;
Pluggable database altered.

CDBTEST> ALTER PLUGGABLE DATABASE pdb_hotclone REFRESH;
Pluggable database altered.

CDBTEST> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;
Pluggable database altered.

CDBTEST> select distinct label from source_user.bigtab;

LABEL

DATA
CDBTEST>
```

11. Drop the current refreshable copy PDB and re-create it to configure it as an automatic refreshable clone.

```
CDBTEST> ALTER PLUGGABLE DATABASE pdb_hotclone CLOSE;
Pluggable database altered.

CDBTEST> ALTER SESSION SET container = CDB$ROOT;
Session altered.

CDBTEST> select pdb_name, status from cdb_pdbs;

PDB_NAME STATUS

PDB_HOTCLONE REFRESHING
PDB$SEED NORMAL

CDBTEST> DROP PLUGGABLE DATABASE pdb_hotclone INCLUDING DATAFILES;
Pluggable database dropped.

CDBTEST> CREATE PLUGGABLE DATABASE pdb_hotclone
 FROM pdb_source_for_hotclone@link_pdb_source_for_hotclone
 FILE_NAME_CONVERT=
 ('/u01/app/oracle/oradata/ORCLCDB/pdb_source_for_hotclone',
```

```
'/u01/app/oracle/oradata/CDBTEST/pdb_hotclone')
REFRESH MODE every 2 minutes;
```

Pluggable database created.

CDBTEST>

**Note:** The refreshable copy PDB must be closed in order for refresh to be completed. If it is not closed when automatic refresh is attempted, the refresh will be deferred until the next scheduled refresh.

12. In terminal *ORCLCDB*, update and commit the source data in *ORCL*.

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

10000 rows updated.

```
ORCLCDB> commit;
```

Commit complete.

```
ORCLCDB> select distinct label from source_user.bigtab;
```

```
LABEL
```

```

```

```
DATA2
```

```
ORCLCDB> exit
```

```
...
```

```
$
```

13. In Session *CDBTEST*, check that the data in *PDB\_HOTCLONE* is refreshed.

- a. Verify that the data is refreshed. It is not refreshed.

```
CDBTEST> ALTER SESSION SET container = pdb_hotclone;
```

Session altered.

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

Pluggable database altered.

```
CDBTEST> select distinct label from source_user.bigtab;
```

```
LABEL
```

```
DATA

1 row selected.

CDBTEST>
```

- b. Close PDB\_HOTCLONE.

```
CDBTEST> ALTER PLUGGABLE DATABASE pdb_hotclone CLOSE;

Pluggable database altered.

CDBTEST>
```

- c. After the sleep 120 command has completed, open the PDB and verify that the data is refreshed.

```
CDBTEST> ! sleep 120

CDBTEST> ALTER PLUGGABLE DATABASE pdb_hotclone OPEN READ ONLY;

Pluggable database altered.

CDBTEST> SELECT DISTINCT label FROM source_user.bigtab;

LABEL

DATA2

1 row selected.

CDBTEST> exit
...
$
```

14. Execute the \$HOME/labs/PDB/cleanup\_hotclones.sh script to drop the PDB\_SOURCE\_FOR\_HOTCLONE in orclcdb and PDB\_HOTCLONE in CDBTEST.

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

15. Close all terminal windows.

## Practice 13-2: Relocating PDBs

---

In this practice, you will move PDB1 from ORCLCDB into CDBTEST in one step, using the Near-zero Downtime PDB Relocation feature.

### Assumptions:

- It is assumed that the database and listener are running. You can use the `pgrep -lf smon` command to verify that the database is started and the `pgrep -lf tns` command to verify that the listener is started. If you need to restart the database and listener, use the `dbstart.sh` script.
- The CDBTEST instance exists and is started.

### Tasks

1. Open a terminal and execute the `setup_pdb3.sh` shell script to re-create ORCLPDB3. This script also adds the PDB3 service name in the `tnsnames.ora` file. You will be prompted for the system password (which will not be shown). See *Course Practice Environment: Security Credentials* for passwords.

```
$ $HOME/labs/DBMod_PDBs/setup_pdb3.sh
...
Input the SYSTEM user password: password
...
$
```

2. In the current terminal window, set the title to *Session ORCLCDB1*. Set the SQL prompt to ORCLCDB1. Verify that the source, `orclcdb`, is configured to use local undo.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / AS SYSDBA
...
SQL> SET sqlprompt "ORCLCDB1> "

ORCLCDB1> SELECT property_name, property_value
 FROM database_properties
 WHERE property_name = 'LOCAL_UNDO_ENABLED';

PROPERTY_NAME PROPERTY_VALUE

LOCAL_UNDO_ENABLED TRUE
SQL>
```

3. Verify that the test user has been created in the ORCLPDB3 PDB, and the table test.bigtab has been populated. See *Course Practice Environment: Security Credentials* for passwords.

**Note:** The net service name is PDB3.

```
ORCLCDB1 > connect test@pdb3
Enter password: password
Connected.

ORCLCDB1> select label, count(*) from test.bigtab group by label;

LABEL COUNT(*)

DATA FROM test.bigtab 10000

ORCLCDB1>
```

4. In Session ORCLCDB1, prepare to relocate ORCLPDB3 from orclcdb into CDBTEST as PDB\_RELOCATED.

- a. In orclcdb, in Session ORCLCDB 1, create the database link to access CDBTEST. See *Course Practice Environment: Security Credentials* for passwords.

```
ORCLCDB1 > connect / as sysdba
Connected.

ORCLCDB1 > drop public database link link_cdbtest;
DROP PUBLIC DATABASE LINK link_CDBTEST
*
ERROR at line 1:
ORA-02024: database link not found

ORCLCDB1 > create public database link link_cdbtest
 connect to system identified by password
 using 'CDBTEST';

Database link created.

ORCLCDB1>
```

- b. List the PDBs. The con\_id for the PDBs may vary from the values shown below.

```
ORCLCDB1 > show pdbs

CON_ID CON_NAME OPEN MODE RESTRICTED

2 PDB$SEED READ ONLY NO
3 ORCLPDB1 READ WRITE NO
4 ORCLPDB2 READ WRITE NO
6 ORCLPDB3 READ WRITE NO
```

```
ORCLCDB1>
```

- c. If ORCLPDB1 or any of the pluggable databases are not open, issue the following command to open all the pluggable databases.

```
ORCLCDB1> alter pluggable database all open;
```

```
Pluggable database altered.
```

```
ORCLCDB1>
```

- d. Re-display the status of all the pluggable databases

```
ORCLCDB1> show pdbs
```

| CON_ID | CON_NAME  | OPEN | MODE  | RESTRICTED |
|--------|-----------|------|-------|------------|
| 2      | PDB\$SEED | READ | ONLY  | NO         |
| 3      | ORCLPDB1  | READ | WRITE | NO         |
| 4      | ORCLPDB2  | READ | WRITE | NO         |
| 5      | ORCLPDB3  | READ | WRITE | NO         |

```
ORCLCDB1> alter pluggable database all save state;
```

```
Pluggable database altered.
```

```
ORCLCDB1>
```

- 5. Open a new terminal, and set the title to *Session CDBTEST*.

- a. Start SQL\*Plus connected to the CDBTEST instance, and set the SQL prompt to CDBTEST.

```
$. oraenv
ORACLE_SID = [orclcdb] ? CDBTEST
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL> set sqlprompt "CDBTEST> "
CDBTEST>
```

- b. In Session *CDBTEST*, create the database link to access ORCLPDB3 in orclcdb. See *Course Practice Environment: Security Credentials* for passwords.

```
CDBTEST> drop public database link link_orclcdb;
DROP PUBLIC DATABASE LINK link_ORCLCDB
*
ERROR at line 1:
ORA-02024: database link not found

CDBTEST> create public database link link_orclcdb
 connect to system identified by password
 using 'ORCLCDB';

Database link created.

CDBTEST>
```

- c. Relocate ORCLPDB3. Display the status of the new PDB.

```
CDBTEST> ! mkdir -p
/u01/app/oracle/oradata/CDBTEST/pdb_relocated

CDBTEST> create pluggable database pdb_relocated
 from orclpdb3@link_orclcdb relocate
 file_name_convert=
(''/u01/app/oracle/oradata/ORCLCDB/orclpdb3',
 '/u01/app/oracle/oradata/CDBTEST/pdb_relocated');

*
ERROR at line 1:
ORA-17628: Oracle error 1031 returned by remote Oracle server
ORA-01031: insufficient privileges

CDBTEST>
```

6. In Session *ORCLCDB1*, grant the required privilege to SYSTEM.

```
ORCLCDB1> grant sysoper to system container=all;

Grant succeeded.

CDBTEST>
```

7. In Session *CDBTEST*, relocate PDB\_RELOCATED from ORCLCDB into CDBTEST.

**Note:** You can relocate with the AVAILABILITY MAX clause, which ensures smooth migration of workload and persistent connection forwarding from ORCLCDB to CDBTEST.

The “maximum availability” mode reduces application impact by handling the migration of connections. The source PDB is preserved in mount state to guarantee the connection

forwarding of the listener to the remote listener where the PDB is now relocated. This forwarding persists even after the relocation operation has been completed and effectively allows for no changes to connect strings. It is expected that connect strings are updated at a time that is convenient for the application. Once this is done and all clients connect to the new host without forwarding, the source PDB can be dropped.

```
CDBTEST> create pluggable database pdb_relocated
 from orclpdb3@link_orclcdb relocate
 file_name_convert=
 ('/u01/app/oracle/oradata/ORCLCDB/orclpdb3',
 '/u01/app/oracle/oradata/CDBTEST/pdb_relocated');

Pluggable database created.

CDBTEST> select pdb_name, status from cdb_pdbs;

PDB_NAME STATUS

PDB_RELOCATED RELOCATING
PDB$SEED NORMAL

CDBTEST>
```

8. Open a terminal window, set the terminal title to *Session ORCLCDB2*. Prepare to start a session and a transaction in *ORCLPDB3* to show that while PDB relocation is taking place the transaction will be transferred to the new relocated PDB.

**DO NOT start** the `$HOME/labs/PDB/sessions.sh` shell script until `CREATE PLUGGABLE DATABASE pdb_relocated` has the RELOCATING status in the terminal window, named *Session CDBTEST*.

9. In *Session ORCLCDB2*, execute `$HOME/labs/PDB/sessions.sh`. It will last around 5000 seconds. Do not wait for the script to complete, continue to the next steps.

```
$ $HOME/labs/DBMod_PDBs/sessions.sh
The Oracle base remains unchanged with value /u01/app/oracle
Input the SYSTEM user password: password
...
SQL> begin
 2 for i in 1..5000 loop
 3 insert into test.bigtab values ('NEW DATA during relocation');
 4 dbms_lock.sleep(1);
 5 commit;
 6 end loop;
 7 end;
 8 /
...
...
```

10. During **sessions.sh** execution:

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

```
ORCLCDB1> select pdb_name, status from cdb_pdbs;

PDB_NAME STATUS

ORCLPDB1 NORMAL
PDB$SEED NORMAL
ORCLPDB2 NORMAL
ORCLPDB3 NORMAL

ORCLCDB1>
```

- In Session **CDBTEST**, you can open the relocated PDB in read-only mode.

Note: the value for NEW DATA during relocation may vary from the example.

```
CDBTEST> alter pluggable database pdb_relocated open read only;

Pluggable database altered.

CDBTEST> alter session set container = pdb_relocated;

Session altered.

CDBTEST> select label, count(*) from test.bigtab group by label;

LABEL COUNT(*)

DATA FROM test.bigtab 10000
NEW DATA during relocation 102

CDBTEST>
```

11. If you consider that **sessions.sh** execution is taking too long, you can open the relocated PDB in force mode. When the newly created PDB is opened in read-write mode for the first time, the final steps of the relocation take effect.

- The source PDB is closed and dropped from the source CDB.
- Any session that was established while the PDB was first opened in read-only mode is preserved if the FORCE option is used to transition the PDB from read-only to read-write.

```
CDBTEST> alter session set container = cdb$root;

Session altered.

CDBTEST> alter pluggable database pdb_relocated
```

```

open read write force;

Pluggable database altered.

CDBTEST>

```

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

```

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

12. Still in Session CDBTEST, verify that the application data is relocated in PDB \_RELOCATED in CDBTEST.

**Note:** the value for NEW DATA during relocation may vary from the example.

```

CDBTEST> alter session set container = pdb_relocated;

Session altered.

CDBTEST> select label, count(*) from test.bigtab group by label;

LABEL COUNT(*)

DATA FROM test.bigtab 10000
NEW DATA during relocation 221

CDBTEST> select pdb_name, status from cdb_pdbs;

PDB_NAME STATUS

PDB_RELOCATED NORMAL

CDBTEST>
```

13. In the Session ORCLCDB 1, verify that orclpdb3 does not exist in ORCLCDB anymore.

```
ORCLCDB1> select pdb_name, status from cdb_pdbs;

PDB_NAME STATUS

ORCLPDB1 NORMAL
PDB$SEED NORMAL
ORCLPDB2 NORMAL

ORCLCDB1> exit
...
$
```

14. In Session CDBTEST, drop PDB\_RELOCATED in CDBTEST.

```
CDBTEST> alter session set container = cdb$root;

Session altered.

CDBTEST> alter pluggable database pdb_relocated close;

Pluggable database altered.

CDBTEST> drop pluggable database pdb_relocated including datafiles;

Pluggable database dropped.

CDBTEST> exit
...
$
```

15. In Session ORCLCDB 1, revoke the SYSOPER privilege from SYSTEM.

```
$ sqlplus / as sysdba
SQL> revoke sysoper from system container = all;

Revoke succeeded.

SQL> exit
...
$
```

16. Close all open terminals.



# **Practices for Lesson 14: Managing PDBs**

## **Practices for Lesson 14: Overview**

---

### **Practices Overview**

In these practices, you will learn how to rename a PDB. You will also investigate the impact of initialization parameter changes.

## Practice 14-1: Renaming a PDB

### Overview

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

### Assumptions

- It is assumed that the database and listener are running. You can use the `pgrep -lf smon` command to verify that the database is started and the `pgrep -lf tns` command to verify that the listener is started. If you need to restart the database and listener, use the `dbstart.sh` script.

### Tasks

Rename the pluggable database name for ORCLPDB3 to PDB3\_ORCL in ORCLCDB. For this purpose, you must open the PDB in RESTRICTED mode.

- Execute the `$HOME/labs/DBMod_PDBs/setup_pdb3.sh` shell script to re-create ORCLPDB3 in ORCLCDB. See *Course Practice Environment: Security Credentials* for passwords.

```
$ $HOME/labs/DBMod_PDBs/setup_pdb3.sh
...
Input the SYSTEM user password: password
...
$
```

Connect to orclcdb root container and list the PDBs.

**Note:** The con\_id values may vary from the output shown below.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base has been set to /u01/app/oracle
$ sqlplus / as sysdba
...
SQL> show pdbs
```

| CON_ID | CON_NAME  | OPEN MODE  | RESTRICTED |
|--------|-----------|------------|------------|
| 2      | PDB\$SEED | READ ONLY  | NO         |
| 3      | ORCLPDB1  | READ WRITE | NO         |
| 4      | ORCLPDB2  | READ WRITE | NO         |
| 5      | ORCLPDB3  | READ WRITE | NO         |

```
SQL>
```

2. Change the global database name for ORCLPDB3 to PDB3\_ORCL. See *Course Practice Environment: Security Credentials* for the password to use.

```
SQL> alter pluggable database orclpdb3 rename global_name to
 pdb3_orcl;
alter pluggable database ORCLPDB3 RENAME GLOBAL_NAME TO pdb3_ORCL
*
ERROR at line 1:
ORA-65046: operation not allowed from outside a pluggable database

SQL> connect sys@pdb3 as sysdba
Enter password: password
Connected.

SQL> alter pluggable database rename global_name to pdb3_orcl;
ALTER PLUGGABLE DATABASE RENAME GLOBAL_NAME TO pdb1_ORCL
*
ERROR at line 1:
ORA-65045: pluggable database not in a restricted mode

SQL>
```

3. Close ORCLPDB3.

```
SQL> alter pluggable database close immediate;

Pluggable database altered.
```

4. Open ORCLPDB3 in restricted mode and confirm the status.

```
SQL> alter pluggable database open restricted;

Pluggable database altered.

SQL> select con_id, name, open_mode, restricted from v$pdbs;

CON_ID NAME OPEN_MODE RES
----- -----
3 ORCLPDB3 READ WRITE YES

SQL>
```

5. Change the global database name for ORCLPDB3 to PDB3\_ORCL.

```
SQL> alter pluggable database rename global_name to pdb3_orcl;

Pluggable database altered.

SQL> select con_id, name, open_mode, restricted from v$pdbs;

CON_ID NAME OPEN_MODE RES
```

```

3 PDB3_ORCL READ WRITE YES
```

```
SQL>
```

6. Open PDB3\_ORCL.

```
SQL> alter pluggable database close immediate;
```

```
Pluggable database altered.
```

```
SQL> alter pluggable database open;
```

```
Pluggable database altered.
```

```
SQL>
```

7. Verify that PDB3\_ORCL is in READ WRITE mode, and then exit sqlplus.

```
SQL> select con_id, name, open_mode, restricted from v$pdbs;
```

| CON_ID | NAME | OPEN_MODE | RES |
|--------|------|-----------|-----|
|--------|------|-----------|-----|

|   |           |            |    |
|---|-----------|------------|----|
| 3 | PDB3_ORCL | READ WRITE | NO |
|---|-----------|------------|----|

```
SQL> exit
```

```
...
```

```
$
```

8. Add the PDB3\_ORCL service name to the tnsnames.ora file with the script

```
$HOME/labs/DBMod_PDBs/add_pdb3_tns.sh.
```

```
$ $HOME/labs/DBMod_PDBs/add_pdb3_tns.sh
```

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

```
$
```

9. Test the connection to PDB3\_ORCL and exit when complete.

```
$ sqlplus system@pdb3_orcl
```

```
Enter Password: password
```

```
...
```

```
Connected
```

```
SQL> exit
```

```
...
```

```
$
```

## Practice 14-2: Setting Parameter Values for PDBs

### Overview

In this practice, you will investigate the impact of initialization parameter changes on PDBs.

### Tasks

- Not all initialization parameters are modifiable at the PDB level. A modifiable one, OPTIMIZER\_USE\_SQL\_PLAN\_BASELINES, has been chosen for the example so as to show how initialization parameters behave at the PDB and CDB level. Connect to orclcdb.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
Connected.
SQL> select ispdb_modifiable from v$parameter
 where name = 'optimizer_use_sql_plan_baselines';

ISPDB

TRUE
SQL>
```

- Check the current value of OPTIMIZER\_USE\_SQL\_PLAN\_BASELINES.

```
SQL> show parameter optimizer_use_sql_plan_baselines

NAME TYPE VALUE

optimizer_use_sql_plan_baselines boolean TRUE
SQL>
```

- Connect to PDB3\_ORCL in ORCLCDB and check the current value of OPTIMIZER\_USE\_SQL\_PLAN\_BASELINES.

```
SQL> connect sys@pdb3_orcl as sysdba
Enter password: password
Connected.

SQL> show parameter optimizer_use_sql_plan_baselines

NAME TYPE VALUE

optimizer_use_sql_plan_baselines boolean TRUE
```

```
SQL>
```

4. Change the parameter value to FALSE in PDB3\_ORCL.

```
SQL> alter system set optimizer_use_sql_plan_baselines=false
 scope=both;
System altered.
```

```
SQL> show parameter optimizer_use_sql_plan_baselines
```

| NAME                             | TYPE    | VALUE |
|----------------------------------|---------|-------|
| optimizer_use_sql_plan_baselines | boolean | FALSE |

```
SQL>
```

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

- Create PDB named test, and open it. See *Course Practice Environment: Security Credentials* for password to use.

```
SQL> connect / as sysdba
Connected.
SQL> ! mkdir -p /u01/app/oracle/oradata/ORCLCDB/test

SQL> create pluggable database test admin user admin
 identified by password roles=(connect)
 create_file_dest='/u01/app/oracle/oradata/ORCLCDB/test';

Pluggable database created.

SQL> alter pluggable database test open;

Pluggable database altered.

SQL>
```

- Add a service name, test, to the tnsnames file with the script

```
$HOME/labs/DBMod_PDBs/add_test_tns.sh
```

```
SQL> ! $HOME/labs/DBMod_PDBs/add_test_tns.sh
The Oracle base remains unchanged with value /u01/app/oracle

SQL>
```

- Connect to the test PDB as SYSBDA.

```
SQL> connect sys@test as sysdba
Enter password: password
Connected.
```

```

SQL> show parameter optimizer_use_sql_plan_baselines

NAME TYPE VALUE

optimizer_use_sql_plan_baselines boolean TRUE
SQL>

```

6. Close and open PDB3\_ORCL.

```

SQL> connect sys@pdb3_orcl as sysdba
Enter password: password
Connected.
SQL> alter pluggable database close immediate;

Pluggable database altered.

SQL> alter pluggable database open;

Pluggable database altered.

SQL> show parameter optimizer_use_sql_plan_baselines

NAME TYPE VALUE

optimizer_use_sql_plan_baselines boolean FALSE
SQL>

```

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

a. Shut down and restart the database instance.

```

SQL> connect / as sysdba
Connected.
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> startup
ORACLE instance started.
...
Database mounted.
Database opened.
SQL> col value format a30
SQL> select con_id, value from v$system_parameter

```

```

where name ='optimizer_use_sql_plan_baselines';

CON_ID VALUE

0 TRUE

SQL>

```

- b. Open the PDBs.

**Note:** The value of con\_id may vary from the values shown below.

```

SQL> alter pluggable database all open;

Pluggable database altered.

SQL> show pdbs

CON_ID CON_NAME OPEN MODE RESTRICTED

2 PDB$SEED READ ONLY NO
3 ORCLPDB1 READ WRITE NO
4 ORCLPDB2 READ WRITE NO
5 ORCLPDB3 READ WRITE NO
6 TEST READ WRITE NO

SQL> select con_id, value from v$system_parameter where name
='optimizer_use_sql_plan_baselines';

CON_ID VALUE

0 TRUE
5 FALSE

SQL>

```

8. Drop PDBs, test and PDB3\_ORCL.

```

SQL> show pdbs

CON_ID CON_NAME OPEN MODE RESTRICTED

2 PDB$SEED READ ONLY NO
6 PDB3_ORCL READ WRITE NO
4 ORCLPDB2 READ WRITE NO
5 TEST READ WRITE NO
3 ORCLPDB1 READ WRITE NO

```

```
SQL> alter pluggable database test close;
Pluggable database altered.

SQL> alter pluggable database pdb3_orcl close;
Pluggable database altered.

SQL> drop pluggable database test including datafiles;

SQL> drop pluggable database pdb3_orcl including datafiles;

SQL> ! rm -rf $ORACLE_BASE/oradata/ORCLCDB/test

SQL> ! rm -rf $ORACLE_BASE/oradata/ORCLCDB/PDB3_ORCL

SQL>
```

9. Confirm PDBs were dropped with a `show pdbs` command then exit SQL\*Plus.

```
SQL> show pdbs

CON_ID CON_NAME OPEN MODE RESTRICTED
----- -----
 2 PDB$SEED READ ONLY NO
 3 ORCLPDB1 READ WRITE NO
 4 ORCLPDB2 READ WRITE NO

SQL> exit
...
$
```

10. Exit all terminals.

# **Practices for Lesson 15: Database Storage Overview**

## **Practices for Lesson 15: Overview**

---

There are no practices for Lesson 15.

# **Practices for Lesson 16: Creating and Managing Tablespaces**

## **Practices for Lesson 16: Overview**

---

### **Overview**

In these practices, you will view information about tablespaces and create new tablespaces.

## Practice 16-1: Viewing Tablespace Information

---

### Overview

In this practice, you use SQL\*Plus to query various views to learn about tablespace content in ORCLPDB1. You also view tablespace information with SQL\*Developer.

### Assumptions

You are logged in as the `oracle` user.

### Tasks

1. Open a new terminal window and source the `oraenv` script.

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

2. Grant DBA to PDBADMIN in ORCLPDB1

- a. Start SQL\*Plus and connect as a sysdba user to ORCLPDB1. Refer to *Practice Environment: Security Credentials* for the password value.

```
$ sqlplus sys/password@ORCLPDB1 as sysdba
...
SQL>
```

- b. Grant DBA to PDBADMIN.

```
SQL> grant dba to pdbadmin;
Grant succeeded.
```

**Note:** Run following query if PDB is not open:

```
SQL> ALTER PLUGGABLE DATABASE ORCLPDB1 OPEN;
```

3. Connect to ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
SQL> connect pdbadmin/password@orclpdb1
Grant succeeded.

SQL>
```

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

| SQL> <b>describe dba_tablespaces</b> |          |                |
|--------------------------------------|----------|----------------|
| 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 ORCLPDB1.

```
SQL> col tablespace_name format a20
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> col contents format a15
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 ALLOCATIO
----- ----- ----- NO NO LOCAL SYSTEM
ONLINE PERMANENT LOGGING
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, ORCLPDB1 is container ID 3. The container ID will vary, depending on the number of times the PDB has been recreated.

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

```
SQL> col table_name format a20
SQL> select table_name from all_tables where
 tablespace_name='USERS' and owner='HR' order by 1;

TABLE_NAME

DEPARTMENTS
EMPLOYEES
JOBS
JOB_HISTORY
LOCATIONS
REGIONS

6 rows selected.

SQL>
```

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

```
SQL> col index_name format a25
SQL> select index_name from all_indexes where
tablespace_name='USERS' and owner='HR' order by 1;

INDEX_NAME

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

19 rows selected.

SQL>
```

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

```
SQL> describe dba_data_files
Name Null? Type

FILE_NAME VARCHAR2 (513)
FILE_ID NUMBER
TABLESPACE_NAME VARCHAR2 (30)
BYTES NUMBER
BLOCKS NUMBER
STATUS VARCHAR2 (9)
RELATIVE_FNO NUMBER
```

|                    |              |
|--------------------|--------------|
| AUTOEXTENSIBLE     | VARCHAR2 (3) |
| MAXBYTES           | NUMBER       |
| MAXBLOCKS          | NUMBER       |
| INCREMENT_BY       | NUMBER       |
| USER_BYTES         | NUMBER       |
| USER_BLOCKS        | NUMBER       |
| ONLINE_STATUS      | VARCHAR2 (7) |
| LOST_WRITE_PROTECT | VARCHAR2 (7) |
| SQL>               |              |

13. List data file information for the SYSAUX tablespace by querying various columns in the DBA\_DATA\_FILES view.

```
SQL> column file_name format a50
SQL> select file_name, autoextensible, bytes, maxbytes, user_bytes
from dba_data_files where tablespace_name='SYSAUX';

FILE_NAME AUT

BYTES MAXBYTES USER_BYTES

/u01/app/oracle/oradata/ORCLCDB/orclpdb1/sysaux01.dbf YES
387973120 3.4360E+10 386924544

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 there are in the SYSAUX tablespace by querying the DBA\_SEGMENTS view. This number will vary.

```
SQL> select count(segment_name) from dba_segments where
tablespace_name='SYSAUX';

COUNT(SEGMENT_NAME)

1317
SQL>
```

15. Find out which index in the SYSAUX tablespace takes up the most space by querying the DBA\_SEGMENTS view. This number of bytes may vary. 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 *
 from (select segment_name, segment_type, bytes
 from dba_segments
 where segment_type = 'INDEX' and
 tablespace_name = 'SYSAUX'
 order by bytes desc)
 WHERE rownum < 2;

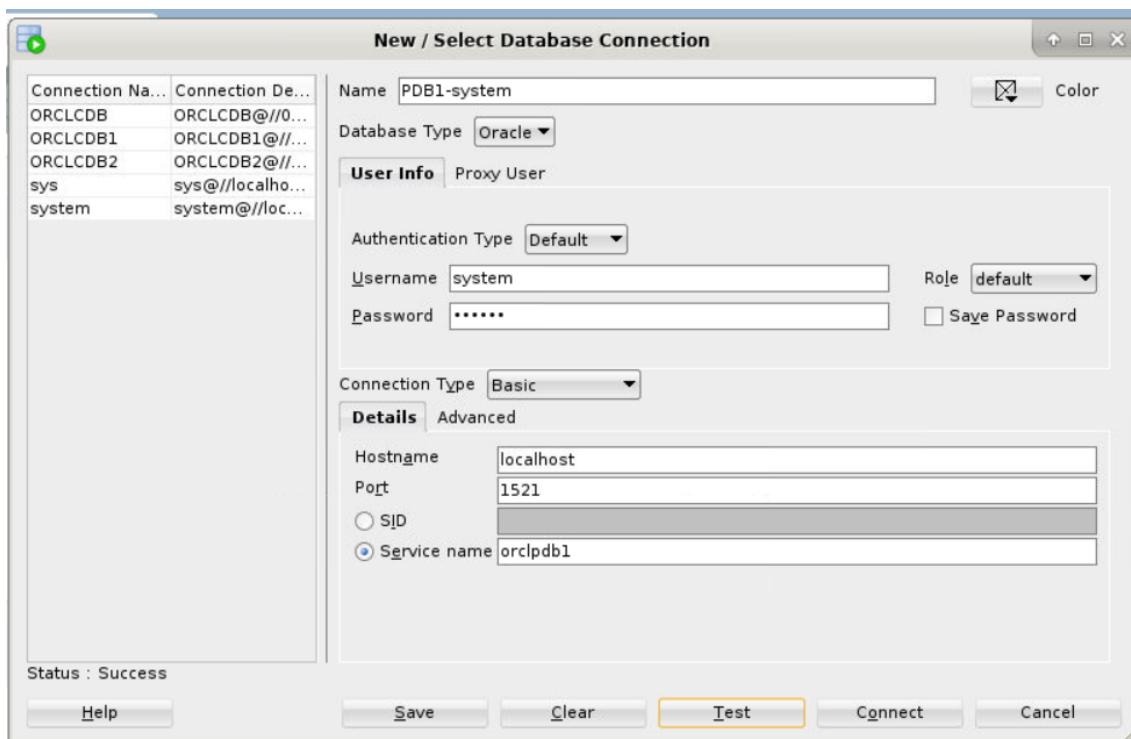
SEGMENT_NAME SEGMENT_TYPE BYTES
----- -----
I_WRI$_OPTSTAT_H_OBJ#_ICOL#_ST INDEX 3145728
SQL>
```

16. Exit SQL\*Plus.

```
SQL> exit
...
$
```

## Viewing Tablespace Information by SQL\*Developer

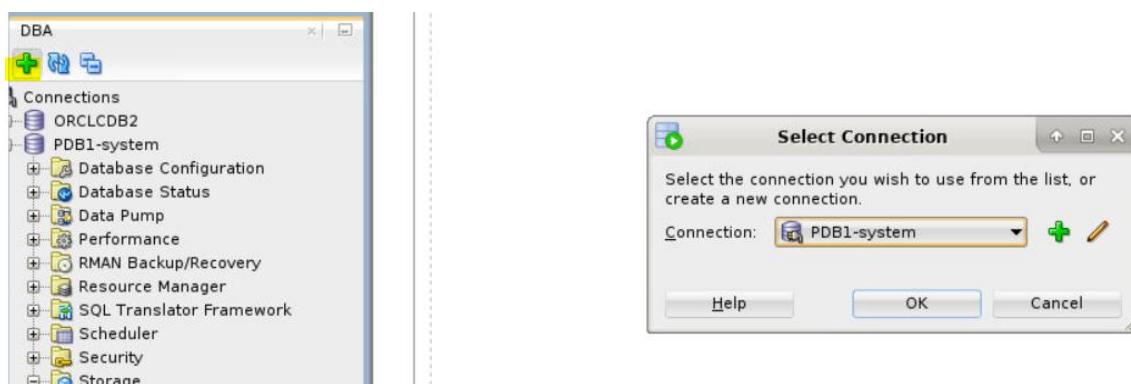
1. Launch SQL\*Developer. Click create **New Connection** and add following details:



2. Expand **PDB1-system** in the Connections pane.

3. In the DBA view, Expand **PDB1-system**.

**Note:** If you do not see the DBA panel on the lower left under Connections & Reports panel, then select **View > DBA**



4. Expand **Storage** and then select **Tablespaces**. A new tab named Tablespaces should appear.

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Team, Tools, Window, and Help. The top navigation bar has three tabs: PDB1-system, PDB1-sys, and Tablespaces, with Tablespaces being the active tab. The left sidebar contains a 'Connections' section with PDB1-sys and PDB1-system expanded, showing Tables (Filtered), Views, and Indexes. Below this is the 'DBA' section, which is expanded and shows Performance, RMAN Backup/Recovery, Resource Manager, SQL Translator Framework, Scheduler, Security, Storage (with Archive Logs, Control Files, Datafiles, Redo Log Groups, Rollback Segments, and Tablespace highlighted with a red arrow), Temporary Tablespace Group, Tuning, and PDB1-system. The main workspace displays a table titled 'Tablespaces' with the following data:

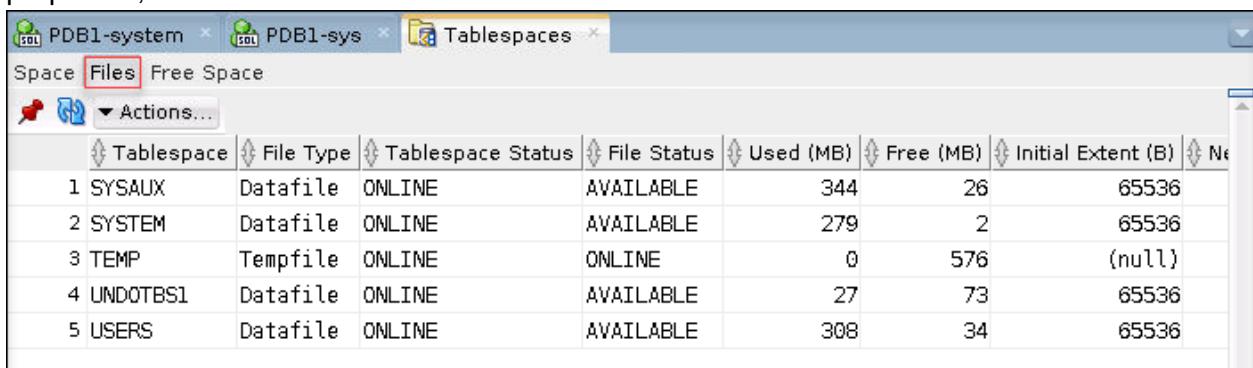
| Tablespace Name | Allocated (MB) | Free (MB) | Used (MB) | % Free | % Used | Max. Bytes (MB) |
|-----------------|----------------|-----------|-----------|--------|--------|-----------------|
| 1 UNDOTBS1      | 100            | 73        | 27        | 73     | 27     | 32768           |
| 2 SYSTEM        | 280            | 2         | 279       | 1      | 99     | 32768           |
| 3 USERS         | 343            | 34        | 308       | 10     | 90     | 32768           |
| 4 SYSAUX        | 370            | 26        | 344       | 7      | 93     | 32768           |
| 5 TEMP          | 576            | 576       | 0         | 100    | 0      | 524288          |

5. All the tablespaces in the **Space** tab are listed with their size, amount of free space, amount used (MB), %Free, %Used, and Maximum Size setting.

The screenshot shows the Oracle SQL Developer interface with the Space tab selected in the top navigation bar. The DBA tree on the left is identical to the previous screenshot. The main workspace displays a table titled 'Tablespaces' with the same data as the previous screenshot:

| Tablespace Name | Allocated (MB) | Free (MB) | Used (MB) | % Free | % Used | Max. Bytes (MB) |
|-----------------|----------------|-----------|-----------|--------|--------|-----------------|
| 1 UNDOTBS1      | 100            | 73        | 27        | 73     | 27     | 32768           |
| 2 SYSTEM        | 280            | 2         | 279       | 1      | 99     | 32768           |
| 3 USERS         | 343            | 34        | 308       | 10     | 90     | 32768           |
| 4 SYSAUX        | 370            | 26        | 344       | 7      | 93     | 32768           |
| 5 TEMP          | 576            | 576       | 0         | 100    | 0      | 524288          |

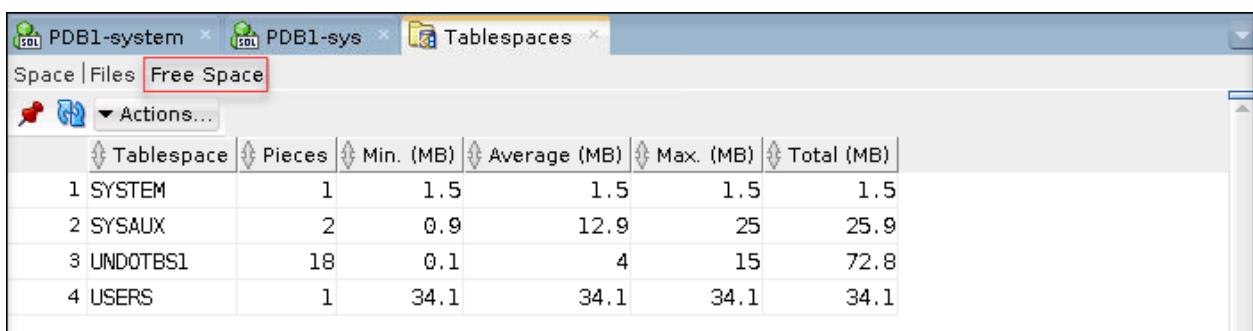
6. In the **Files** tab, File Type, Tablespace Status, File Status, Used (MB), Free (MB), other properties, and Datafile Name are listed.



The screenshot shows the SQL Developer interface with three tabs at the top: PDB1-system, PDB1-sys, and Tablespaces. The Tablespaces tab is selected. Below it, there are two tabs: Space and Files. The Files tab is selected and highlighted with a red box. A toolbar below the tabs includes icons for refresh, search, and actions. A table displays information about five tablespaces:

|   | Tablespace | File Type | Tablespace Status | File Status | Used (MB) | Free (MB) | Initial Extent (B) | Next Extent (B) |
|---|------------|-----------|-------------------|-------------|-----------|-----------|--------------------|-----------------|
| 1 | SYSAUX     | Datafile  | ONLINE            | AVAILABLE   | 344       | 26        | 65536              | 65536           |
| 2 | SYSTEM     | Datafile  | ONLINE            | AVAILABLE   | 279       | 2         | 65536              | 65536           |
| 3 | TEMP       | Tempfile  | ONLINE            | ONLINE      | 0         | 576       | (null)             | 65536           |
| 4 | UNDOTBS1   | Datafile  | ONLINE            | AVAILABLE   | 27        | 73        | 65536              | 65536           |
| 5 | USERS      | Datafile  | ONLINE            | AVAILABLE   | 308       | 34        | 65536              | 65536           |

7. In the **Free Space** tab, Pieces, Min(MB), Average (MB), Max (MB) and Total (MB) are listed.



The screenshot shows the SQL Developer interface with three tabs at the top: PDB1-system, PDB1-sys, and Tablespaces. The Tablespaces tab is selected. Below it, there are two tabs: Space and Free Space. The Free Space tab is selected and highlighted with a red box. A toolbar below the tabs includes icons for refresh, search, and actions. A table displays information about four tablespaces:

|   | Tablespace | Pieces | Min. (MB) | Average (MB) | Max. (MB) | Total (MB) |
|---|------------|--------|-----------|--------------|-----------|------------|
| 1 | SYSTEM     | 1      | 1.5       | 1.5          | 1.5       | 1.5        |
| 2 | SYSAUX     | 2      | 0.9       | 12.9         | 25        | 25.9       |
| 3 | UNDOTBS1   | 18     | 0.1       | 4            | 15        | 72.8       |
| 4 | USERS      | 1      | 34.1      | 34.1         | 34.1      | 34.1       |

8. Question: In this example, how much of the SYSAUX tablespace is used?

Answer: 94% of the SYSAUX tablespace has been used. It has 25MB of free space left.

**Note:** The values in your database may differ from what is shown in this example.

9. Close SQL\*Developer.

## Practice 16-2: Creating a Tablespace

---

### Overview

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

### Assumptions

You are logged in 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. Set the environment variable for the ORCLCDB database, then start SQL\*Plus and connect to ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus pdbadmin/password@orclpdb1
...
SQL>
```

2. Execute the `CreateINVENTORystablespace.sql` script.

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

SQL> CREATE SMALLFILE TABLESPACE INVENTORY
 2 DATAFILE
 3 '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF'
SIZE 5242880
 4 DEFAULT NOCOMPRESS
 5 ONLINE
 6 SEGMENT SPACE MANAGEMENT AUTO
 7 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/DBMod_Storage/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;

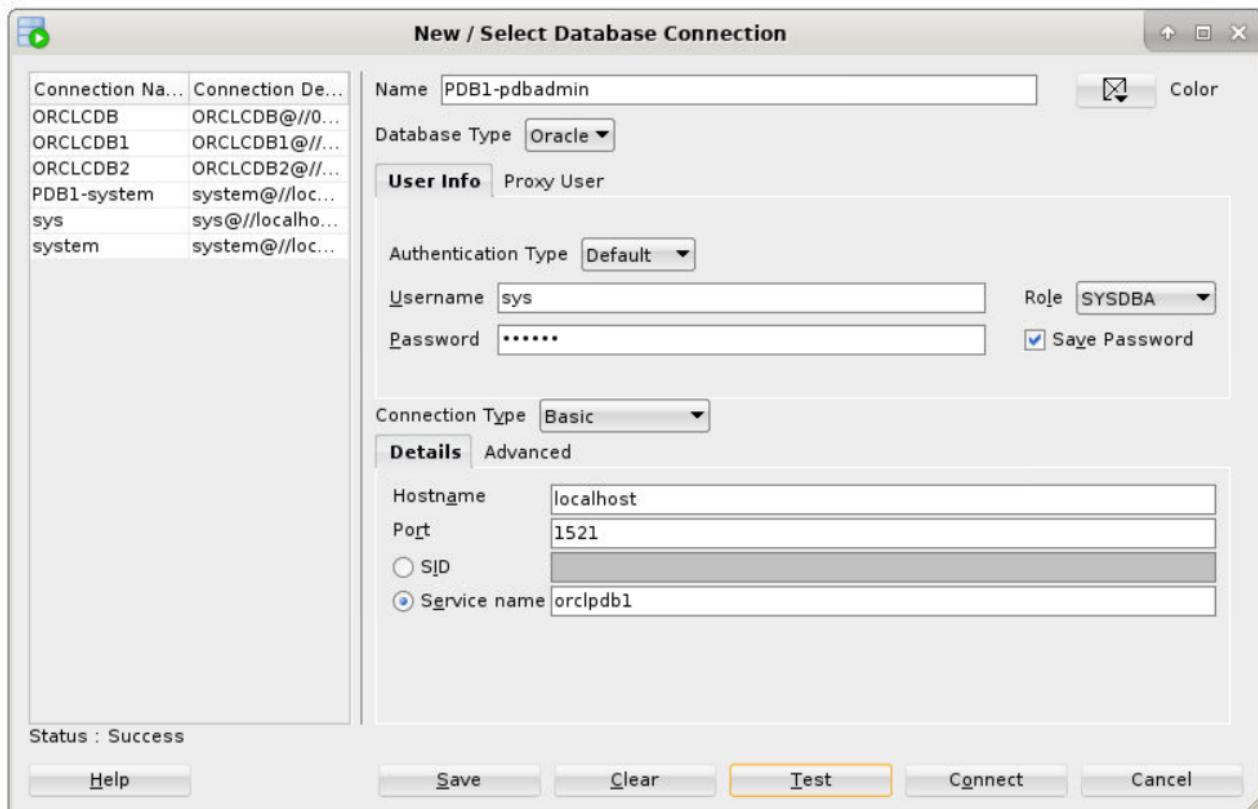
Commit complete.
```

```
SQL> quit
...
$
```

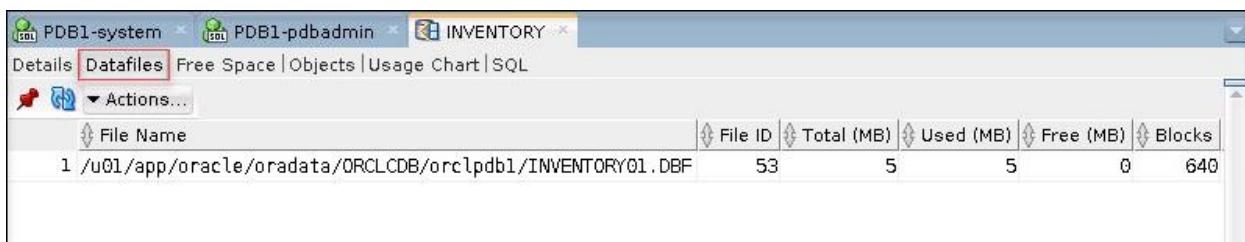
### Use SQL\*Developer to Increase the Size of the INVENTORY01.DBF Data File

Fix the problem that you encountered in the previous section by increasing the size of the INVENTORY01.dbf data file. Use SQL\*Developer because it provides an easy-to-use interface when working with tablespaces.

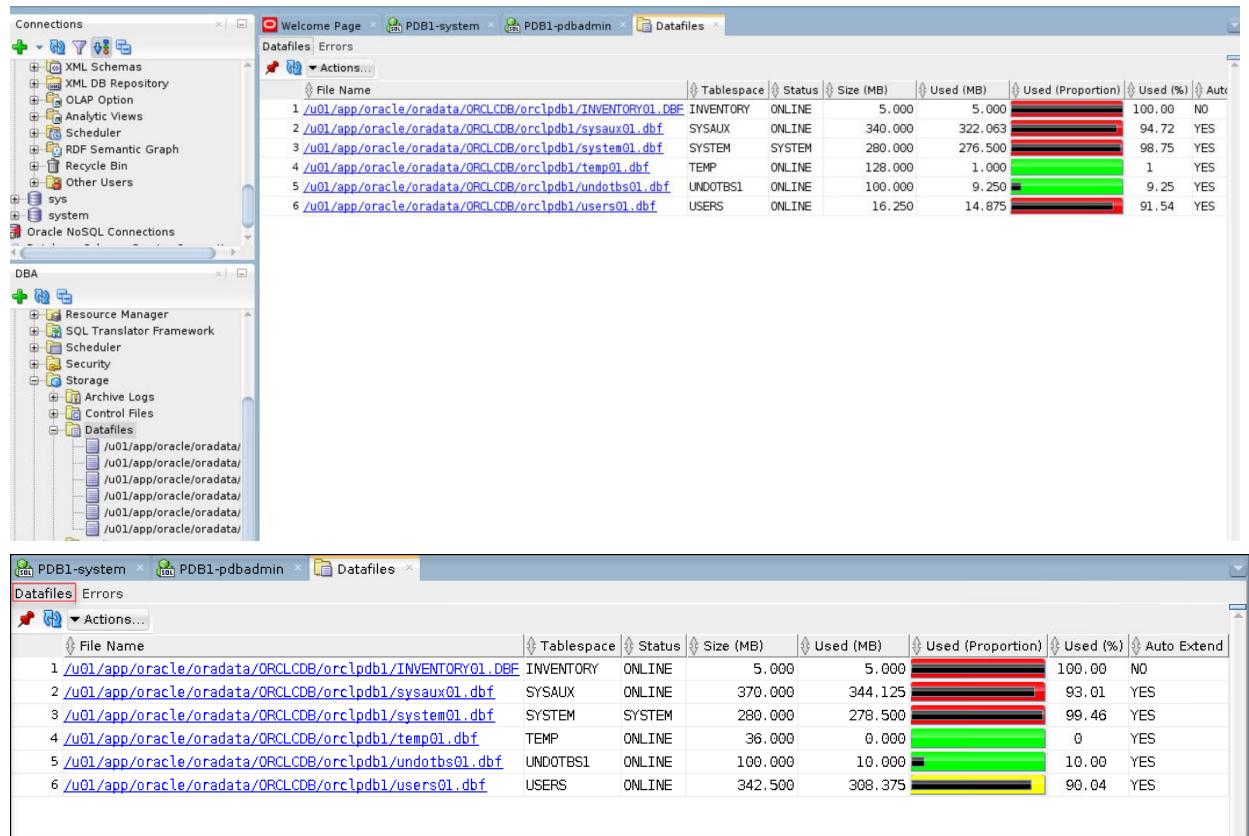
10. Launch SQL\*Developer. Click create **New Connection** and add following details:



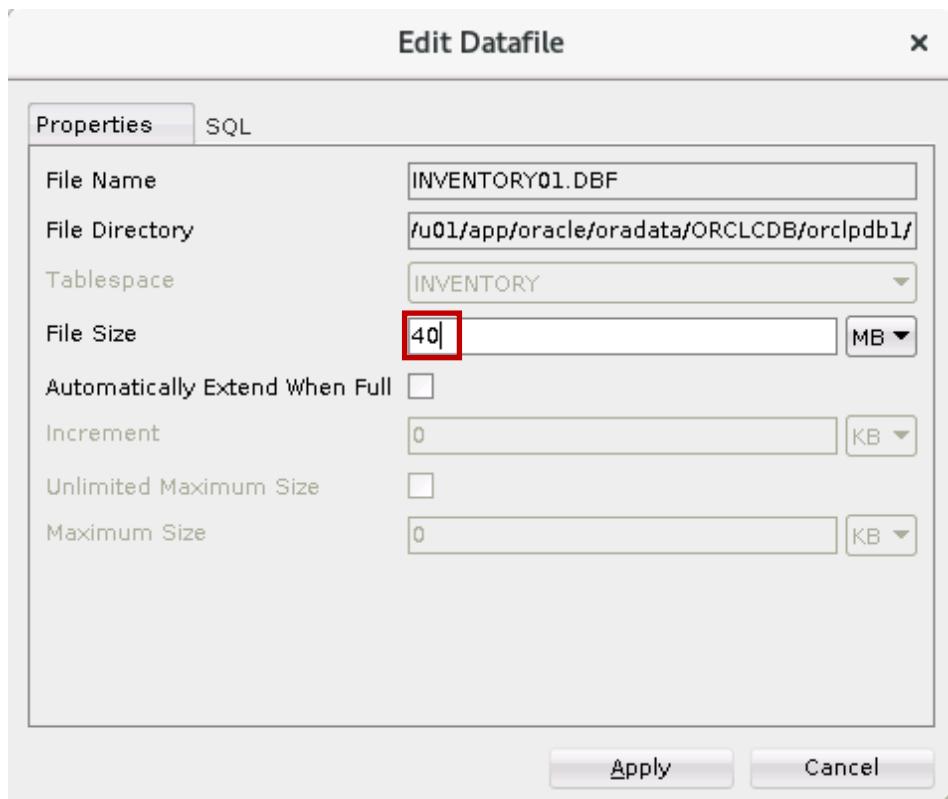
1. In the DBA panel, expand **PDB1-pdbadmin**
2. Expand **Storage** and then select **Tablespaces**.
3. Double click the **INVENTORY** tablespace and select the **Datafiles** tab.
4. Now that you have found the name of **INVENTORY01.DBF** data file.



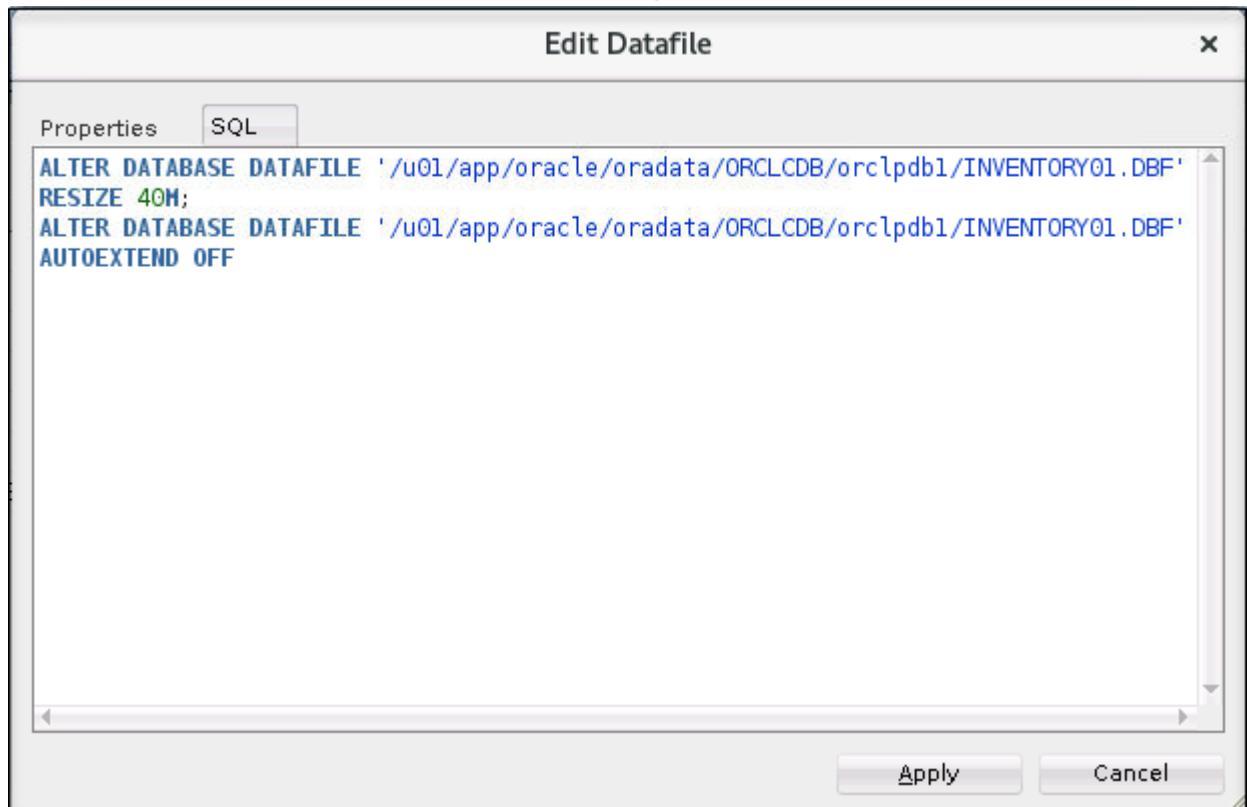
5. In the DBA pane, select **Datafiles**. You can see that `INVENTORY01.DBF` datafile is 100% used.



6. Double click the `INVENTORY01.DBF` filename. Click **Actions > Edit**.  
 7. In the Edit Datafile box, enter a File Size of **40M**. Don't click Apply just yet.



8. Click the **SQL** tab to view the SQL command that performs the resize action.



9. In the dialog box, click **Apply**.
10. In the Successful dialog box, click **OK**. Data file has been successfully resized.

11. Verify that the change is reflected in the SQL\*Developer interface. The size for the INVENTORY tablespace should now be set to 40MB.

The screenshot shows the SQL\*Developer interface with two tabs open: 'PDB1-system' and 'PDB1-pdbadmin'. The 'Actions...' button is selected. A table displays the following properties for the INVENTORY tablespace:

| Name                     | Value                                                    |
|--------------------------|----------------------------------------------------------|
| 1 Name                   | /u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF |
| 2 Tablespace             | INVENTORY                                                |
| 3 Status                 | ONLINE                                                   |
| 4 File Size (MB)         | 40.00                                                    |
| 5 Auto Extend            | NO                                                       |
| 6 Increment (MB)         | 0.00                                                     |
| 7 Maximum File Size (MB) | 0.00                                                     |

### Use SQL\*Developer to Add a Data File to the INVENTORY Tablespace

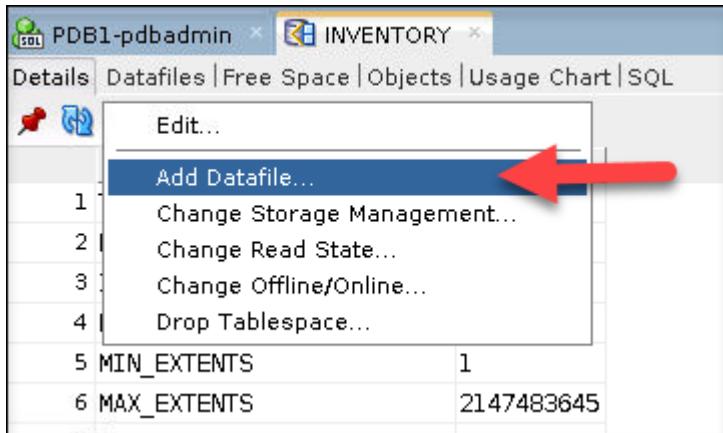
1. In the DBA pane, Expand Storage, and click Tablespaces

The screenshot shows the DBA pane with the 'Tables' tab selected. Under 'Space', the 'Tables' tab is also selected. The table lists the following tablespaces:

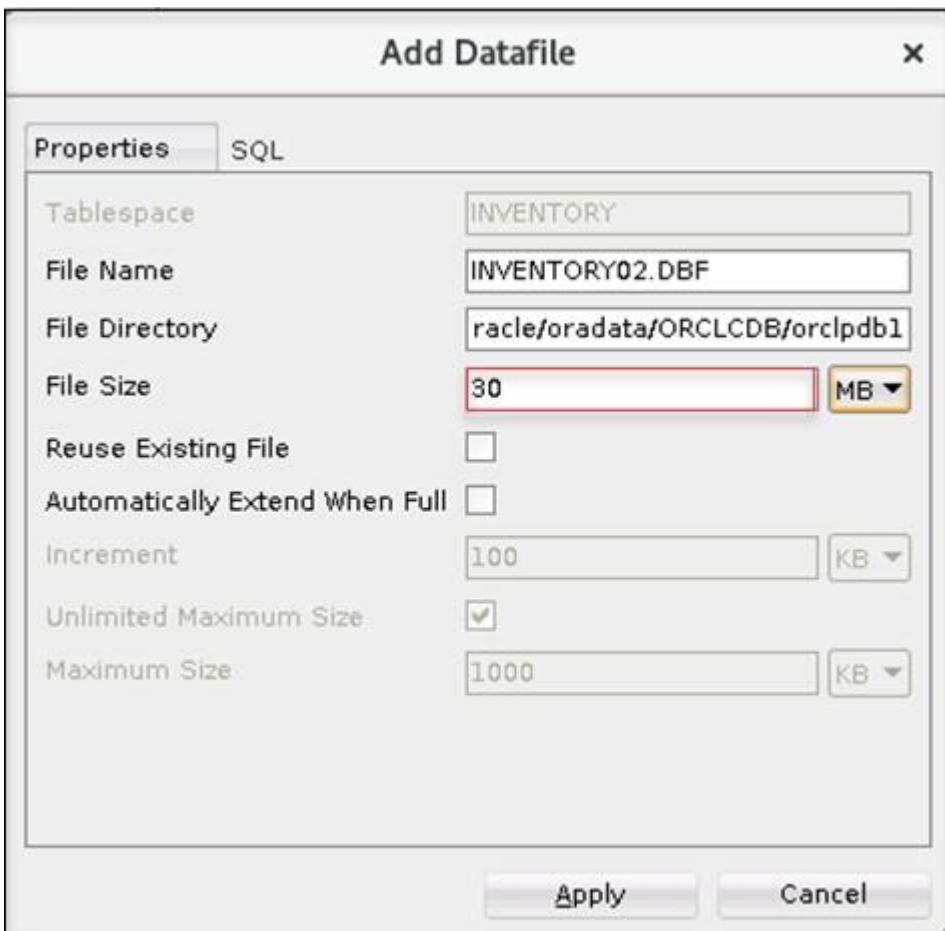
| Tablespace Name | Allocated (MB) | Free (MB) | Used (MB) | % Free | % Used | Max. Bytes (MB) |
|-----------------|----------------|-----------|-----------|--------|--------|-----------------|
| 1 INVENTORY     | 40             | 35        | 5         | 88     | 12     | 40              |
| 2 UNDOTBS1      | 100            | 19        | 81        | 19     | 81     | 32768           |
| 3 SYSTEM        | 280            | 1         | 279       | 0      | 100    | 32768           |
| 4 USERS         | 360            | 52        | 308       | 14     | 86     | 32768           |
| 5 SYSAUX        | 370            | 25        | 345       | 7      | 93     | 32768           |
| 6 TEMP          | 576            | 576       | 0         | 100    | 0      | 524288          |

2. Double click the INVENTORY tablespace.

3. Expand **Actions** and then select **Add Datafile**.



4. The "Add Datafiles" dialog box is displayed.
- Enter File Name: **INVENTORY02.DBF**
  - Enter File Directory: **/u01/app/oracle/oradata/ORCLCDB/orclpdb1/**
  - Enter the File Size: **30M**.



- Click **SQL** tab and view the SQL code being generated.
- Click **Apply**.
- In the Successful window, click **OK**

- Refresh the **INVENTORY** tab by clicking on the Datafiles subtab and verify that it now has two data files: **INVENTORY01.DBF** and **INVENTORY02.DBF**

| File Name                                                  | File ID | Total (MB) | Used (MB) | Free (MB) | Blocks | Autoextensible |
|------------------------------------------------------------|---------|------------|-----------|-----------|--------|----------------|
| 1 /u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY02.DBF | 54      | 30         | 1         | 29        | 3840   | NO             |
| 2 /u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF | 53      | 40         | 5         | 35        | 5120   | NO             |

- Close the SQL\*Developer window.

### 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.

- Return to your terminal window.

- Start SQL\*Plus and connect to ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the password value

```
$ sqlplus pdbadmin/password@orclpdb1
...
SQL>
```

- Run the `CreateTable_X.sql` script, located in `/home/oracle/labs/DBMod_Storage`. The script runs without any errors.

```
SQL> @/home/oracle/labs/DBMod_Storage/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
...
$
```

13. Start SQL\*Plus again and connect to ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the password value.

```
$ sqlplus pdbadmin/password@orclpdb1
...
SQL>
```

14. 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

15. Drop the INVENTORY tablespace.

```
SQL> drop tablespace inventory including contents and datafiles;

Tablespace dropped.

SQL>
```

16. Exit SQL\*Plus.

```
SQL> exit
...
$
```

17. Close the terminal session.

## Practice 16-3: Managing Temporary and Permanent Tablespaces

---

### Overview

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

### Assumptions

- The PDB, ORCLPDB1, exists and is open.

### Tasks

- Run following command as root user: **chmod -R 777 /home/oracle/labs/**
- Then execute the `$HOME/labs/storage/glogin_6`. This script sets formatting for all columns selected in queries.

```
$ su - oracle
$ $HOME/labs/DBMod_Storage/glogin_6.sh
```

- View permanent and temporary tablespaces properties in ORCLCDB.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
...
$ sqlplus / as sysdba
...
SQL> select property_name, property_value
 from database_properties
 where property_name like 'DEFAULT_%TABLE%';

----- -----
PROPERTY_NAME PROPERTY_VALUE

DEFAULT_PERMANENT_TABLESPACE USERS
DEFAULT_TEMP_TABLESPACE TEMP

SQL> select tablespace_name, con_id from cdb_tablespaces;

SYSTEM 1
SYSAUX 1
UNDOTBS1 1
TEMP 1
USERS 1
SYSTEM 3
SYSAUX 3
```

```
UNDOTBS1 3
TEMP 3
USERS 3
SYSTEM 4
SYSAUX 4
UNDOTBS1 4
TEMP 4
USERS 4
```

15 rows selected.

```
SQL> select tablespace_name, con_id from cdb_tablespaces
 where tablespace_name like 'TEMP%' order by 2;
```

```
TABLESPACE_NAME CON_ID

TEMP 1
TEMP 3
TEMP 4
```

```
SQL>
```

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

```
SQL> create tablespace cdata
 datafile '/u01/app/oracle/oradata/ORCLCDB/cdata_01.dbf'
 size 10m;
```

Tablespace created.

```
SQL> select tablespace_name, con_id from cdb_tablespaces
 where tablespace_name = 'CDATA' ;
```

```
TABLESPAC CON_ID

CDATA 1

SQL>
```

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

```
SQL> alter database default tablespace cdata;
```

Database altered.

```
SQL> select property_name, property_value
```

```

from database_properties
where property_name like 'DEFAULT_%TABLE%';

PROPERTY_NAME PROPERTY_VALUE

DEFAULT_PERMANENT_TABLESPACE CDATA
DEFAULT_TEMP_TABLESPACE TEMP

SQL>

```

6. Create a permanent tablespace, LDATA, in ORCLPDB1. Refer to *Course Practice Environment: Security Credentials* for the password value.

```

SQL> connect system@orclpdb1
Enter password: password
Connected.

SQL> create tablespace ldata datafile
 '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/ldata_01.dbf' size 10m;

Tablespace created.

SQL>

```

7. Make the LDATA tablespace the default tablespace in the ORCLPDB1 container.

```

SQL> alter pluggable database default tablespace ldata;

Pluggable database altered.

SQL> select property_name, property_value
 from database_properties
 where property_name like 'DEFAULT_%TABLE%';

PROPERTY_NAME PROPERTY_VALUE

DEFAULT_PERMANENT_TABLESPACE LDATA
DEFAULT_TEMP_TABLESPACE TEMP

SQL>

```

8. Create a temporary tablespace in the CDB root. Refer to *Course Practice Environment: Security Credentials* for the password value.

```

SQL> CONNECT system
Enter password: password
Connected.

```

```
SQL> create temporary tablespace temp_root tempfile
'/u01/app/oracle/oradata/ORCLCDB/temproot_01.dbf' size 500m;
```

Tablespace created.

```
SQL>
```

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

```
SQL> alter database default temporary tablespace temp_root;
```

Database altered.

```
SQL> select property_name, property_value
 from database_properties
 where property_name like 'DEFAULT_%TABLE%';
```

| PROPERTY_NAME                | PROPERTY_VALUE |
|------------------------------|----------------|
| DEFAULT_PERMANENT_TABLESPACE | CDATA          |
| DEFAULT_TEMP_TABLESPACE      | TEMP_ROOT      |

```
SQL>
```

10. Create a temporary tablespace TEMP\_PDB1 in ORCLPDB1. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect system@orclpdb1
```

Enter password: **password**

Connected.

```
SQL> create temporary tablespace temp_pdb1 tempfile
'/u01/app/oracle/oradata/ORCLCDB/orclpdb1/temp pdb1_01.dbf' size
100m;
```

Tablespace created.

```
SQL>
```

11. Make TEMP\_PDB1 the default temporary tablespace in ORCLPDB1.

```
SQL> alter database default temporary tablespace temp_pdb1;
```

Database altered.

```
SQL> select property_name, property_value
 from database_properties
 where property_name like 'DEFAULT_%TABLE%';
```

| PROPERTY_NAME                | PROPERTY_VALUE |
|------------------------------|----------------|
| DEFAULT_PERMANENT_TABLESPACE | ldata          |
| DEFAULT_TEMP_TABLESPACE      | temp_pdb1      |
| SQL>                         |                |

12. Create a temporary tablespace MYTEMP in ORCLPDB1.

```
SQL> create temporary tablespace my_temp tempfile
 '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/mytemp_01.dbf' size 10m;
Tablespace created.

SQL>
```

13. Display default tablespaces of another PDB in ORCLCDB. Create a new PDB using the \$HOME/labs/DBMod\_Storage/setup\_newpdb.sql SQL script. This script creates a new PDB, queries it for the default tablespaces, and then drops the PDB.

```
SQL> Connect / as sysdba
Connected.

SQL> @$HOME/labs/DBMod_Storage/setup_newpdb.sql
...
SQL> CREATE PLUGGABLE DATABASE newpdb ADMIN USER admin
 IDENTIFIED BY fenago ROLES=(CONNECT)
 CREATE_FILE_DEST='/u01/app/oracle/oradata/ORCLCDB/newpdb';

Pluggable database created.

SQL> alter PLUGGABLE DATABASE newpdb open;

Pluggable database altered.

SQL> CONNECT system/fenago@//localhost:1521/newpdb

SQL> set echo on
SQL> SELECT property_name, property_value
 2 FROM database_properties
 3 WHERE property_name LIKE 'DEFAULT_%TABLE%';

PROPERTY_NAME PROPERTY_VALUE

DEFAULT_PERMANENT_TABLESPACE SYSTEM
DEFAULT_TEMP_TABLESPACE TEMP

SQL>
```

14. Manage default permanent and temporary tablespaces of users.

- a. Create a common user C##U. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

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

SQL> create user c##u identified by password;
User created.

SQL>
```

- b. View the default tablespace and temporary tablespace assignment for user C##U in all containers.

```
SQL> select username, default_tablespace,
 temporary_tablespace, con_id
 from cdb_users
 where username = 'C##U'
 order by 4;

USERNAME DEFAULT_TABLESPACE TEMPORARY_TABLESPACE CON_ID
----- -----
C##U CDATA TEMP_ROOT 1
C##U LDATA TEMP_PDB1 3
C##U USERS TEMP 4
C##U SYSTEM TEMP 5

SQL>
```

- c. Create a local user LU in ORCLPDB1. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect system@orclpdb1
Enter password: password
Connected.

SQL> create user lu identified by password;
User created.

SQL>
```

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

```
SQL> select username, default_tablespace, temporary_tablespace
 from dba_users
 where username = 'LU';

USERNAME DEFAULT_TABLESPACE TEMPORARY_TABLESPACE
----- -----
LU LDATA TEMP_PDB1

SQL>
```

- e. Change the temporary tablespace assignment for user LU to MY\_TEMP in ORCLPDB2.

```
SQL> alter user lu temporary tablespace my_temp;
User altered.

SQL>
```

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

```
SQL> select username, default_tablespace, temporary_tablespace
 from dba_users
 where username = 'LU';

USERNAME DEFAULT_TABLESPACE TEMPORARY_TABLESPACE
----- -----
LU LDATA MY_TEMP

SQL>
```

15. Log out of SQL\*Plus.

```
SQL> exit
...
$
```

16. Close all terminals.

# **Practices for Lesson 17: Improving Space Usage**



## **Practices for Lesson 17: 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.

### **Time Estimate:**

It is estimated that this practice can be completed in 40 minutes.

## Practice 17-1: 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 alert at both thresholds. The alerts notify you and often provide recommendations on how to resolve the reported problem.

### Tasks

#### Set a Warning Threshold

1. Source the `oraenv` script.

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

2. Connect to `ORCLPDB1` as the `SYSTEM` user. Refer to *Course Practice Environment: Security Credentials* for the `password` value.

```
$ sqlplus system/password@orclpdb1
...
SQL>
```

3. Execute the `DBMS_SERVER_ALERT.SET_THRESHOLD` procedure to reset the database-wide threshold values for the Tablespace Space Usage metric.

**Note:** The following command must either be all on one line or each line must be ended with a '-' character with no spaces following it.

```
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>
```

4. 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>
```

5. In ORCLPDB1, 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 ORCLPDB1

```
SQL> alter session set container = orclpdb1;

Session altered.

SQL>
```

- b. Create the `TBSALERT` tablespace by executing the `Create_TBSALERT_TS.sql` script.

```
SQL> set echo on
SQL> @/home/oracle/labs/DBMod_Storage/Create_TBSALERT_TS.sql
SQL> CREATE TABLESPACE tbsalert
2 DATAFILE
'/u01/app/oracle/oradata/ORCLCDB/orclpdb1/tbsalert.dbf'
3 SIZE 120M REUSE LOGGING EXTENT MANAGEMENT LOCAL
4 SEGMENT SPACE MANAGEMENT AUTO;
```

Tablespace created.

SQL>

6. 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/DBMod_Storage/TBSALERT_free_space.sql
SQL> select df.tablespace_name tablespace, fs.bytes free,
 df.bytes , fs.bytes *100/ df.bytes pct_free
 from dba_data_files df ,dba_free_space fs
 where df.tablespace_name = fs.tablespace_name
 and df.tablespace_name = 'TBSALERT';
```

| TABLESPACE | FREE      | BYTES     | PCT_FREE   |
|------------|-----------|-----------|------------|
| TBSALERT   | 124780544 | 125829120 | 99.1666667 |

SQL>

7. 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>

8. 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>
```

9. 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 |         |

```
REASON RESOLUT

Threshold is updated on metrics "Tablespace Space Usage" cleared
SQL>
```

10. Exit SQL\*Plus.

```
SQL> exit
...
$
```

11. Execute the \$HOME/labs/seg\_advsr\_setup.sh shell script to create and populate new tables in the TBSALERT tablespace.

```
$ $HOME/labs/DBMod_Storage/seg_advsr_setup.sh
...
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
...
$
```

12. Check the fullness level of the TBSALERT tablespace to see if the warning level has been reached.

- a. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$ sqlplus system/password@orclpdb1
...
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/DBMod\_Storage/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/DBMod_Storage/TBSALERT_free_space.sql
SQL> SELECT df.tablespace_name tablespace, fs.bytes free,
 df.bytes, fs.bytes *100/ df.bytes PCT_FREE
 FROM dba_data_files df, dba_free_space fs
 WHERE df.tablespace_name = fs.tablespace_name
 AND df.tablespace_name = 'TBSALERT';

TABLESPACE FREE BYTES PCT_FREE
----- -----
TBSALERT 49283072 125829120 39.1666667

SQL>
```

- d. Wait a few minutes, then 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. You may have to wait as much as 10 minutes.

```
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@ORCLPDB1] 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.

13. 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>
```

14. Wait a few minutes.

15. Query the fullness of the tablespace. The result shows that the tablespace is 75% full.

```
SQL> SELECT sum(bytes) * 100 / 125829120
 FROM dba_extents
 WHERE tablespace_name='TBSALERT';

SUM(BYTES)*100/125829120

75
SQL>
```

16. 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. *It may take as long as 10 minutes to display 75 percent.*

```
SQL> SELECT reason FROM dba_outstanding_alerts WHERE
object_name='TBSALERT';
```

REASON

---

Tablespace [TBSALERT@PDB1] is [75 percent] full

SQL>

17. 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>

18. 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
 from dba_extents
 where tablespace_name='TBSALERT';
```

SUM(BYTES)\*100/125829120

---

```
SQL>
```

### Create a Segment Advisor Task.

19. 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/DBMod_Storage/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>
```

20. Execute the task.

```
SQL> DECLARE
 tname VARCHAR2(128) := 'my_seg_task';
BEGIN
 dbms_advisor.EXECUTE_TASK(tname);
END;
/
```

PL/SQL procedure successfully completed.

```
SQL>
```

21. 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>
```

22. Execute the \$HOME/labs/DBMod\_Storage/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> @/home/oracle/labs/DBMod_Storage/segments_to_shrink.sql

SQL> col attr1 format a5
SQL> col attr2 format a12
SQL> col message format a55
SQL> set echo on
SQL> SELECT attr1, attr2, message FROM dba_advisor_findings f,
 dba_advisor_objects o 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 EMPLOYEES2 Perform shrink, estimated savings is 18873242
bytes.
HR EMPLOYEES1 Perform shrink, estimated savings is 18873242
bytes.
HR EMPLOYEES3 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>
```

23. 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>
```

24. 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
 FROM dba_extents
 WHERE tablespace_name='TBSALERT';

SUM(BYTES)*100/125829120

30.15625

SQL>
```

25. Drop the TBSALERT tablespace.

```
SQL> DROP TABLESPACE tbsalert INCLUDING CONTENTS AND DATAFILES;

Tablespace dropped.

SQL>
```

26. Exit SQL\*Plus.

```
SQL> exit
...
$
```

27. Close all terminals.

## Practice 17-2: 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. Open a terminal and set the environment variable to ORCLCDB database.

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

2. Execute the `$HOME/labs/DBMod_Storage/setup_index.sh` shell script to create an index with low compression on the `HR.TEST` table in ORCLPDB1.

```
$ $HOME/labs/DBMod_Storage/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
...
$
```

- Start SQL\*Plus and connect to ORCLPDB1 as the HR user. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$ sqlplus hr/password@orclpdb1
...
SQL>
```

- 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>
```

- 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>
```

- Exit SQL\*Plus, but keep the terminal window open.

```
SQL> exit
...
$
```

- View the different compression levels that exist in your Oracle Database version. To do this, use the cat command to review the predefined SQL script that creates the DBMS\_COMPRESSION package.

```
$ less $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
...
create or replace package dbms_compression authid current_user is

COMP_NOCOMPRESS CONSTANT NUMBER := 1;
COMP_ADVANCED CONSTANT NUMBER := 2;
COMP_QUERY_HIGH CONSTANT NUMBER := 4;
COMP_QUERY_LOW CONSTANT NUMBER := 8;
COMP_ARCHIVE_HIGH CONSTANT NUMBER := 16;
COMP_ARCHIVE_LOW CONSTANT NUMBER := 32;
COMP_BLOCK CONSTANT NUMBER := 64;
COMP_LOB_HIGH CONSTANT NUMBER := 128;
COMP_LOB_MEDIUM CONSTANT NUMBER := 256;
COMP_LOB_LOW CONSTANT NUMBER := 512;
COMP_INDEX_ADVANCED_HIGH CONSTANT NUMBER := 1024;
COMP_INDEX_ADVANCED_LOW CONSTANT NUMBER := 2048;
COMP_BASIC CONSTANT NUMBER := 4096;
COMP_INMEMORY_NOCOMPRESS CONSTANT NUMBER := 8192;
COMP_INMEMORY_DML CONSTANT NUMBER := 16384;
COMP_INMEMORY_QUERY_LOW CONSTANT NUMBER := 32768;
COMP_INMEMORY_QUERY_HIGH CONSTANT NUMBER := 65536;
COMP_INMEMORY_CAPACITY_LOW CONSTANT NUMBER := 131072;
COMP_INMEMORY_CAPACITY_HIGH CONSTANT NUMBER := 262144;

COMP_RATIO_MINROWS CONSTANT NUMBER := 1000000;
COMP_RATIO_ALLROWS CONSTANT NUMBER := -1;
COMP_RATIO_LOB_MINROWS CONSTANT NUMBER := 1000;
COMP_RATIO_LOB_MAXROWS CONSTANT NUMBER := 5000;
COMP_RATIO_INDEX_MINROWS CONSTANT NUMBER := 100000;

OBJTYPE_TABLE CONSTANT NUMBER := 1;
OBJTYPE_INDEX CONSTANT NUMBER := 2;
OBJTYPE_PART CONSTANT NUMBER := 3;
OBJTYPE_SUBPART CONSTANT NUMBER := 4;
...

grant execute on dbms_compression to public

```

```
/

show errors;

@?/rdbms/admin/sqlsessend.sql

$
```

8. Start SQL\*Plus and connect to ORCLPDB1 as the HR user. Refer to *Course Practice Environment: Security Credentials* for the **password** value.  
Hint: use the up arrow key several times to recall the command from the OS command-line buffer.

```
$ sqlplus hr/password@orclpdb1
...
SQL>
```

9. 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/DBMod\_Storage/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/DBMod_Storage/Compression_index_low.sql
SQL> set serveroutput on
SQL> DECLARE
blkcnt_cmp pls_integer;
blkcnt_uncmp pls_integer;
row_cmp pls_integer;
row_uncmp pls_integer;
cmp_ratio pls_integer;
comptype_str varchar2(100);
BEGIN
DBMS_COMPRESSION.GET_COMPRESSION_RATIO
(
scratchtbsname => 'USERS',
ownname => 'HR',
objname => 'I_TEST',
subobjname => NULL,
comptype => dbms_compression.COMP_INDEX_ADVANCED_LOW,
blkcnt_cmp => blkcnt_cmp,
blkcnt_uncmp => blkcnt_uncmp,
row_cmp => row_cmp,
row_uncmp => row_uncmp,
cmp_ratio => cmp_ratio,
```

```

comptype_str => comptype_str,
subset_numrows => dbms_compression.COMP_RATIO_MINROWS,
objtype => dbms_compression.OBJTYPE_INDEX
);
DBMS_OUTPUT.PUT_LINE('Block used by compressed index = ' || blkcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Block used by uncompressed index = ' || blkcnt_uncmp);
DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
DBMS_OUTPUT.PUT_LINE('Compression ratio org = ' || cmp_ratio);
END;
/

```

Block used by compressed index = 809  
 Block used by uncompressed index = 1029  
 Compression type = "Compress Advanced Low"  
 Compression ratio org = 1

PL/SQL procedure successfully completed.

SQL>

10. 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/DBMod\_Storage/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/DBMod_Storage/Compression_index_high.sql
SQL> set serveroutput on
SQL> DECLARE
blkcnt_cmp pls_integer;
blkcnt_uncmp pls_integer;
row_cmp pls_integer;
row_uncmp pls_integer;
cmp_ratio pls_integer;
comptype_str varchar2(100);
BEGIN
DBMS_COMPRESSION.GET_COMPRESSION_RATIO
(
scratchtbsname => 'USERS',
ownname => 'HR',
objname => 'I_TEST',
subobjname => NULL,

```

```

comptype => dbms_compression.COMP_INDEX_ADVANCED_HIGH,
blkcnt_cmp => blkcnt_cmp,
blkcnt_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>
```

11. 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.
12. Rebuild the index with Advanced High Compression.

```

SQL> alter index hr.i_test rebuild compress advanced high;

Index altered.

SQL>
```

13. 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>
```

14. 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>
```

15. *Question:* Is it possible to revert back to the initial compression level?

*Answer:* Yes.

16. Revert back to the initial compression level.

```
SQL> alter index hr.i_test rebuild nocompress;

Index altered.

SQL>
```

17. 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>
```

18. Exit SQL\*Plus.

```
SQL> exit
...
$
```

## Practice 17-3: 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 `ORCLPDB1` as the `SYSTEM` user. Refer to *Course Practice Environment: Security Credentials* for the `password` value.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus system/password@orclpdb1
...
SQL>
```

2. Grant the PDBADMIN user DBA role.

```
SQL> grant dba to pdbadmin;
```

Grant succeeded.

```
SQL>
```

3. Connect to ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect pdbadmin/password@orclpdb1
```

Connected.

```
...
```

```
SQL>
```

4. Execute the \$HOME/labs/DBMod\_Storage/CreateINVENTORYTablespace.sql script to create an unpopulated tablespace named INVENTORY.

```
SQL> @$HOME/labs/DBMod_Storage/CreateINVENTORYTablespace.sql
```

Tablespace created.

```
SQL>
```

5. Execute the \$HOME/labs/DBMod\_Storage/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/DBMod_Storage/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
...
$
```

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 ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$ sqlplus pdbadmin/password@orclpdb1
...
SQL>
```

8. Enable resumable mode.

```
SQL> alter session enable resumable;
Session altered.

SQL>
```

9. Re-execute the CreateTable\_X script. The script is suspended.

```
SQL> @$HOME/labs/DBMod_Storage/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 ;

SQL>
```

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

12. Open another terminal window. This will be referred to as Window 2.

13. Source the oraenv script.

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

14. Start SQL\*Plus and connect to ORCLPDB1 as SYSTEM. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$ sqlplus system/password@orclpdb1
...
SQL>
```

15. 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 pagesize 100
SQL> column sql_text format a60
SQL> column error_msg format a60
SQL> SELECT status, name, sql_text, error_msg FROM dba_resumable;

STATUS NAME SQL_TEXT

ERROR_MSG

SUSPENDED User PDBADMIN(INSERT INTO x SELECT * FROM x
 105), Session
 40, Instance 1
ORA-01653: unable to extend table PDBADMIN.X by 128 in table
space INVENTORY

SQL>
```

16. Exit SQL\*Plus, but keep the terminal window open.

```
SQL> exit
...
$
```

17. 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.

```
$ tail -30
/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace/alert_orclcdb.log
...
2020-10-21T02:31:45.793234+00:00
ORCLPDB1(3):ORA-1653: unable to extend table PDBADMIN.X by 128 in
tablespace INVENTORY [ORCLPDB1]
2020-10-21T02:33:46.433096+00:00
ORCLPDB1(3):statement in resumable session 'User PDBADMIN(105),
Session 40, Instance 1' was suspended due to
ORCLPDB1(3): ORA-01653: unable to extend table PDBADMIN.X by 128
in tablespace INVENTORY
```

```
$
```

18. Proceed with the appropriate corrective action. Because the INVENTORY tablespace is not autoextensible, you can configure it as autoextensible with a size limit.

- a. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. . Refer to Course Practice Environment: Security Credentials for the **password** value.

```
$ sqlplus system/password@orclpdb1
...
SQL>
```

- b. Query the DBA\_DATA\_FILES view to verify whether the INVENTORY tablespace is autoextensible. The result shows that the tablespace is not.

```
SQL> column file_name format a60
SQL> select file_name, autoextensible from dba_data_files where
tablespace_name='INVENTORY';

FILE_NAME AUT

/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF NO
SQL>
```

- c. Enable autoextend for the INVENTORY01.DBF data file.

```
SQL> alter database datafile
'/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF'
autoextend on maxsize 10m;

Database altered.
SQL>
```

- d. Query the DBA\_DATA\_FILES view again to verify whether the INVENTORY tablespace is autoextensible. The result shows that it is.

```
SQL> select file_name, autoextensible from dba_data_files where
tablespace_name='INVENTORY';

FILE_NAME AUT

/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF YES
SQL>
```

### Window 1: Check the Suspended Session

19. 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
```

...  
\$

20. Close the terminal window.

#### **Window 2: Verify that there are no Suspended Sessions.**

21. 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>
```

22. Exit from SQL\*Plus and close the terminal window.

```
SQL> exit
```

...

```
$ exit
```

# **Practices for Lesson 18: Managing Undo Data**

## **Practices for Lesson 18: Overview**

---

### **Overview**

In these practices, you will view undo activity and configure the database to support twelve-hour retention for flashback operations.

### **Time Estimate**

It is estimated that this practice can be completed in 2 minutes.

## Practice 18-1: Managing Undo Tablespaces in a PDB

---

### Overview

In this practice, you manage undo tablespaces in PDBs.

### Tasks

1. Start SQL\*Plus and connect to the CDB\$ROOT as system. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$. oraenv
ORACLE_SID = [orclcdbs] ? orclcdbs
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus system
Enter password: password
connected to:
...
SQL>
```

2. Display the undo tablespaces used in the CDB.

Note: your FILE# may be different but the TS# and CON\_ID should be the same.

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

 FILE# NAME TS# CON_ID
----- -----
 4 UNDOTBS1 2 1
 8 UNDOTBS1 2 2
 11 UNDOTBS1 2 3
 15 UNDOTBS1 2 4
 49 UNDOTBS1 2 5

SQL>
```

Q: According to the list of undo tablespaces, what can you conclude about the undo mode used?

A: Because there is an undo tablespace in each container, the undo mode used is the local undo mode.

Q2: Why should you use the local undo mode?

A2: The local undo mode is useful for hot cloning, PDB relocation, and PDB proxying.

3. Verify that the undo mode is LOCAL.

```
SQL> SELECT property_name, property_value
 FROM database_properties
 WHERE property_name = 'LOCAL_UNDO_ENABLED';

PROPERTY_NAME PROPERTY_VALUE

LOCAL_UNDO_ENABLED TRUE

SQL>
```

4. Exit from SQL\*Plus.

```
SQL> exit
...
$
```

5. Execute the script \$HOME/labs/DBMod\_Storage/reset\_DBMod\_Storage.sh  
This script will clean up the environment for the following lessons. Note: this script will take several minutes to complete.

```
$ $HOME/labs/DBMod_Storage/reset_DBMod_Storage.sh
...
```

6. Close all open terminals.

# **Practices for Lesson 19: Creating and Managing User Accounts**

## **Practices for Lesson 19: Overview**

---

### **Overview**

In these practices, you will create users and roles. You will grant privileges to users and roles.

## Practice 19-1: 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.
- ORCLPDB1 administrator named `PDB1_ADMIN`: Create this user as a local user in `ORCLPDB1` 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.
- The `ORCLPDB2` pdb exists and is pristine. `ORCLPDB2` will be used to refresh `ORCLPDB1`.

## Tasks

**Reset** ORCLPDB1 and common users.

1. Execute following command as root: **chmod -R 777 /home/oracle/labs**
2. Open terminal and source the environment variables to ORCLCDB

```
$ su - oracle
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
```

3. Execute /home/oracle/labs/DBMod\_UsersSec/reset\_ORCLPDB1.sh.

**Note:** Error messages saying object does not exist can be ignored.

```
$ /home/oracle/labs/DBMod_UsersSec/reset_ORCLPDB1.sh
...
$
```

4. Reset common users.

**Note:** Error messages saying object does not exist can be ignored.

```
$ /home/oracle/labs/DBMod_UsersSec/reset_users_roles.sh
...
$
```

### Create c##CDB\_ADMIN1

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

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

6. 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.

**Note:** The username in the Oracle database is NOT case sensitive, all user names are stored in uppercase. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
SQL> create user c##CDB_ADMIN1 identified by password
 container=all temporary tablespace temp
 account unlock;

User created.

SQL>
```

7. 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>
```

8. **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.

9. 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
CTXSYS
DBSFWUSER
...
SYSTEM
WMSYS
XDB
XS$NULL
```

37 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.

10. Disconnect from the root container.

```
SQL> disconnect
...
SQL>
```

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

```
SQL> show user
USER is ""
SQL>
```

12. Connect to the root container as *anyuser/anystring* and exercise the SYSDBA privilege.

**Note:** If you are connected to the OS as a user that is a privileged database user, that is a member of the DBA group, you can enter anything as a user name and anything as a password and be connected.

```
SQL> connect anyuser/anystring as sysdba
Connected.
SQL>
```

13. Show the current container name.

```
SQL> show con_name

CON_NAME

CDB$ROOT
SQL>
```

14. 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 / user with the SYSDBA privilege, not SYS.

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

15. 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
...
USE ANY SQL TRANSLATION PROFILE
```

```
WRITE ANY ANALYTIC VIEW CACHE

253 rows selected.

SQL>
```

16. Disconnect from the root container.

```
SQL> disconnect
...
SQL>
```

17. Connect to the root container as c##CDB\_ADMIN1 again, but this time, do not exercise the SYSDBA privilege. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect c##cdb_admin1/password
Connected.
SQL>
```

18. 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>
```

19. 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
...
UPDATE ANY TABLE
USE ANY JOB RESOURCE
USE ANY SQL TRANSLATION PROFILE

237 rows selected.

SQL>
```

20. Switch to ORCLPDB1 by issuing the ALTER SESSION command.

```
SQL> alter session set container = orclpdb1;
Session altered.

SQL>
```

21. Show the current container. It is ORCLPDB1.

```
SQL> show con_name

CON_NAME

ORCLPDB1
SQL>
```

### Create the PDB1\_ADMIN User

You just connected to ORCLPDB1 as c##CDB\_ADMIN1. You need to be logged into ORCLPDB1 to create a local administrator for ORCLPDB1. The c##CDB\_ADMIN1 user can create the PDB1\_ADMIN user.

22. Create a local user named PDB1\_ADMIN 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\_ADMIN 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_admin identified by password
 default tablespace users temporary tablespace temp
 account unlock;

User created.

SQL>
```

23. Grant PDB1\_ADMIN the DBA role and the CREATE SESSION privilege in ORCLPDB1 only. This is an example of granting a privilege and role locally.

```
SQL> grant create session, dba to pdb1_admin;

Grant succeeded.

SQL>
```

24. List the local user accounts for ORCLPDB1 by querying the DBA\_USERS view. The PDB1\_ADMIN account is included in the list.

**Note:** The PDBADMIN user was created when ORCLPDB1 was created.

```
SQL> select distinct username from dba_users
 where common='NO' order by username;
```

```
USERNAME

```

```
BI
HR
IX
OE
PDB1_ADMIN
PDBADMIN
PM
SH
```

```
8 rows selected.
```

```
SQL>
```

25. Disconnect c##CDB\_ADMIN1 from PDB1.

```
SQL> disconnect
...
SQL>
```

26. Connect to ORCLPDB1 as PDB1\_ADMIN. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect pdb1_admin/password@orclpdb1
Connected.
SQL>
```

27. Show the current user. You are connected as PDB1\_ADMIN1.

```
SQL> show user
USER is "PDB1_ADMIN"
SQL>
```

28. View the list of privileges for PDB1\_ADMIN 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>
```

29. Try to connect to the root container as the PDB1\_ADMIN user. This user does not have the SET CONTAINER privilege and does not exist in 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>
```

30. Exit SQL\*Plus.

```
SQL> exit
...
$
```

31. Exit all terminals.

## Practice 19-2: Creating a Local User for an Application

---

### Overview

In this practice, you log in to ORCLPDB1 as the local administrator (PDB1\_ADMIN) 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. Open a terminal and use oraenv to set the environment to ORCLCDB. Then start SQL\*Plus and connect to ORCLPDB1 as the PDB1\_ADMIN user. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus pdb1_admin/password@orclpdb1
...
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 ORCLPDB1 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 1;
```

USERNAME

```

BI
HR
INVENTORY
IX
OE
PDB1_ADMIN
PDBADMIN
PM
SH
```

9 rows selected.

```
SQL>
```

### Connect as INVENTORY and Verify Privileges

5. Disconnect PDB1\_ADMIN from ORCLPDB1.

```
SQL> disconnect
...
SQL>
```

6. Verify that the INVENTORY user account can connect to ORCLPDB1. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect inventory/password@orclpdb1
Connected.
SQL>
```

7. 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 1;

PRIVILEGE

CREATE SESSION

SQL>
```

8. Exit SQL\*Plus.

```
SQL> exit
...
$
```

9. Close all terminals.

## Practice 19-3: 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. Open a new terminal & list the file: `/etc/group`

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 groups you select to be the Oracle software owner (`oinstall` by default) and `dba` group must be created in the OS before the software is installed. Notice that these groups are included in the list below. The `dba` group consists of the `oracle` user.

```
$ cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
...
oinstall:x:54321:oracle
dba:x:54322:oracle
oper:x:54323:oracle
backupdba:x:54324:oracle
dgdba:x:54325:oracle
kmdba:x:54326:oracle
racdba:x:54330:oracle

$
```

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.

```
$ whoami
oracle
$
```

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:login_shell`. Passwords are stored in the `/etc/shadow` file, so an `x` is used here as a placeholder.

```
$ grep oracle /etc/passwd
oracle:x:54321:54321::/home/oracle:/bin/bash
$
```

- 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.

```
$ grep oinstall /etc/group
oinstall:x:54321:
$
```

- 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.

```
$ grep oracle /etc/group
oinstall:x:54321:oracle
dba:x:54322:oracle
oper:x:54323:oracle
backupdba:x:54324:oracle
dgdba:x:54325:oracle
kmdba:x:54326:oracle
racdba:x:54330:oracle
$
```

- d. An alternate way to get all of this information for the current user in a single command is the command `id`.

```
$ id
uid=54321(oracle) gid=54321(oinstall)
groups=54321(oinstall),54322(dba),54323(oper),54324(backupdba),54325(dgdba),54326(kmdba),54330(racdba)
$
```

## 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\$PFILE\_USERS view contains information about users that have been granted administrative privileges.

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

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL>
```

5. View the columns in the `V$PFILE_USERS` view by issuing the `DESCRIBE` command.

```
SQL> describe v$pfile_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>
```

6. List the users in the password file by querying the V\$PFILE\_USERS view.

```
SQL> col username format a20
SQL> select username from v$pfile_users;

USERNAME

SYS
C##CDB_ADMIN1

SQL>
```

7. Find out the SYS user's account status and whether the SYS user has the SYSDBA privilege by querying the V\$PFILE\_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$pfile_users where
username='SYS';

ACCOUNT_STATUS SYSDBA

OPEN TRUE

SQL>
```

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

```
SQL> exit
...
$
```

9. Close all terminals.



# **Practices for Lesson 20: Configuring Privilege and Role Authorization**

## **Practices for Lesson 20: Overview**

---

### **Overview**

In these practices, you will create roles. You will grant privileges to roles.

## Practice 20-1: 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 `ORCLPDB1` 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 host machine as the `oracle` user.
- It is assumed that the database and listener are running. You can use the `pgrep -lf smon` command to verify that the database is started and the `pgrep -lf tns` command to verify that the listener has started. If you need to restart the database and listener, use the `dbstart.sh` script.

### Tasks

#### Explore the Privileges and Roles Granted to PDBADMIN

1. Open a new Terminal, set the environment to `ORCLCDB`, 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 `ORCLPDB1`, which you will do in the next step.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ 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 `PDBADMIN` has not been granted any privileges directly. However, there may be privileges granted through roles.

```
SQL> alter session set container=orclpdb1;
Session altered.

SQL> select * from dba_sys_privs where grantee='PDBADMIN' ;
```

```
no rows selected
```

```
SQL>
```

3. List the roles granted to the PDB1\_ADMIN 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> column granted_role format a10
SQL> select granted_role, admin_option from cdb_role_privs where
grantee='PDBADMIN';
```

```
GRANTED_RO ADM

PDB_DBA YES
```

```
SQL>
```

4. List the system privileges granted to the PDB\_DBA role by querying the ROLE\_SYS\_PRIVS view.
- a. 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 ORCLPDB1 as the SYS user, you have access to all role information. The results show that the PDB\_DBA role consists of two system privileges: CREATE SESSION and CREATE PLUGGABLE DATABASE.

```
SQL> select privilege from role_sys_privs where role='PDB_DBA'
order by 1;

PRIVILEGE

CREATE PLUGGABLE DATABASE
CREATE SESSION

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> col granted_role format a15
SQL> select granted_role from dba_role_privs where grantee =
'PDB_DBA' order by 1;

GRANTED_ROLE

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 1;

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

8. Grant the DBA role locally to PDBADMIN.

```
SQL> grant dba to pdbadmin;

Grant succeeded.

SQL>
```

9. 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 1;

GRANTED_RO

DBA
PDB_DBA

SQL>
```

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

```
SQL> exit
...
$
```

11. Close all terminals

## Practice 20-2: Using SQL\*Developer to Create Local Roles

---

### Overview

In this practice, the PDBADMIN user uses SQL\*Developer to create the following local roles in ORCLPDB1:

- 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 3-2 Using SQL Developer to Create Local Users.

### Assumptions

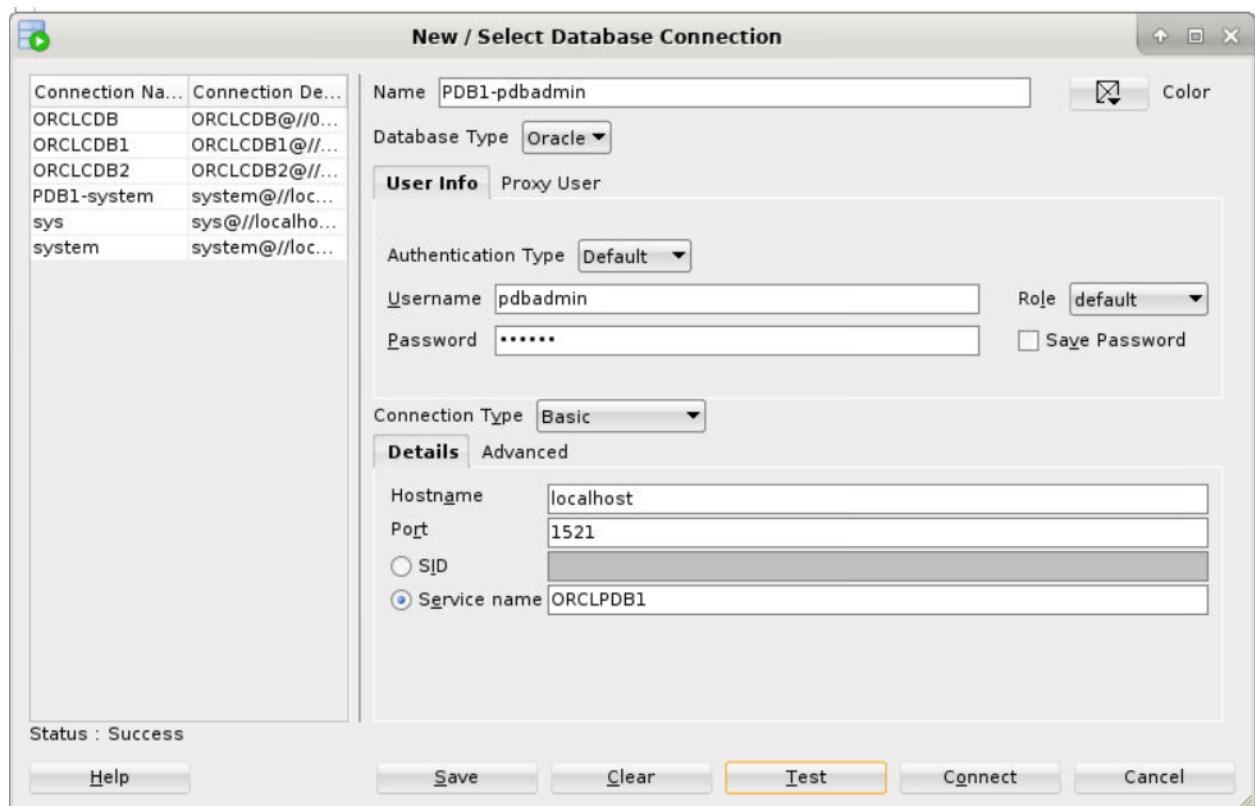
You are currently logged in as the `oracle` OS user.

You completed Practice 2-1 Granting the DBA Role to PDBADMIN.

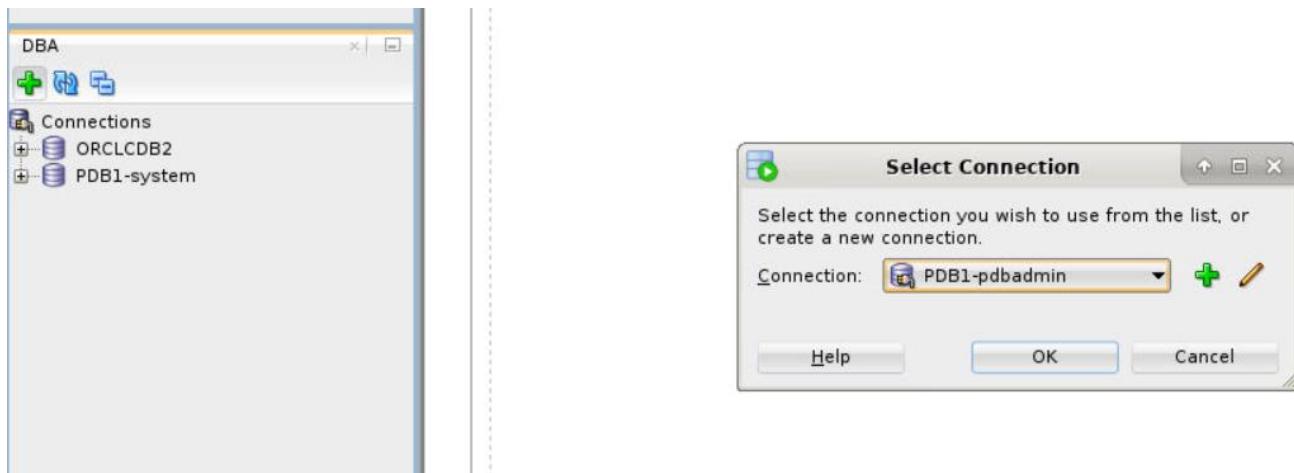
### Tasks

#### Log in to SQL\*Developer (PDB1)

1. Create a connection to ORCLPDB1 as the PDBADMIN.
  - a. Start SQL\*Developer. The SQL\*Developer icon is on the desktop.



- b. In the Connections pane on the left, click the Name: **PDB1-pdbadmin** (a connection for ORCLPDB1 as PDBADMIN)
- c. In the DBA connection box, click the connection: **PDB1-pdbadmin**

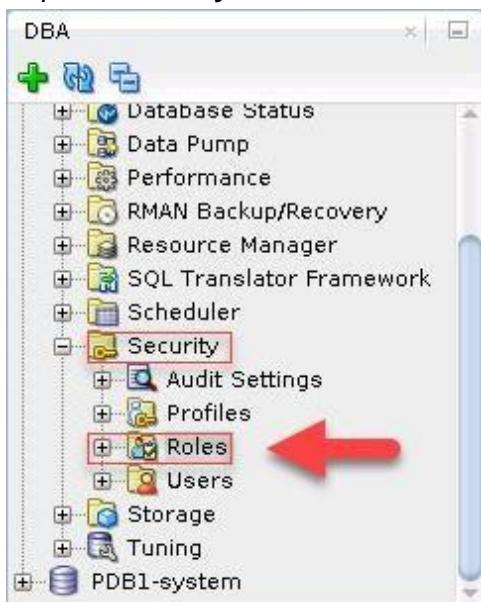


## Create the HRCLERK Role

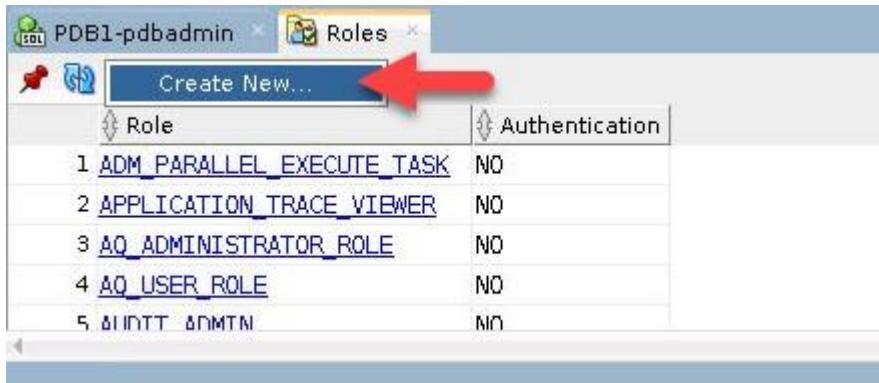
2. Expand the PDB1-pdbadmin connection in the DBA box.



3. Expand **Security** and then select **Roles**.



4. In the Roles tab, select Actions > Create New ...



5. In the Create Role box,
- On the **Roles** tab, enter Role Name: **HRCLERK**.
  - Click the **SQL** tab to view the SQL statement.
  - Click **Apply**.
  - In the Successful box, click **OK**.
  - In the Create Role box, click **Close**.
6. Verify that the **HRCLERK** role is listed in the table.

Note you may have to click on refresh

A screenshot of the Oracle Database Roles window. The title bar shows 'PDB1-pdbadmin' and 'Roles'. Below the title bar, there are two tabs: 'Role' and 'Authentication'. A red arrow points to the 'Role' tab. At the top left of the main area, there is a toolbar with icons for 'Create New...', 'Edit', 'Delete', and 'Actions...'. The 'Actions...' icon is highlighted with a red arrow. Below the toolbar is a table with columns 'Role' and 'Authentication'. The table contains several rows, including 'GSMADMIN\_ROLE', 'GSMROOTUSER\_ROLE', 'GSMUSER\_ROLE', 'GSM\_POOLADMIN\_ROLE', 'HRCLERK', 'HS\_ADMIN\_EXECUTE\_ROLE', 'HS\_ADMIN\_ROLE', 'HS\_ADMIN\_SELECT\_ROLE', 'IMP\_FULL\_DATABASE', 'JAVADEBUGPRIV', 'JAVAIDPRIV', and 'JAVASYSPRIV'. The row for 'HRCLERK' is highlighted with a red box.

## Add Object Privileges to HRCLERK role

7. In the Connections box, expand **PDB1-pdbadmin> Other USERS> HR > Tables**, then click **Employees**.

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Connections' tree view is expanded to show various schemas like DIP, DWF, DV\$SYS, GGSYS, GSMADMIN\_INTERNAL, GSMCATUSER, GSMUSER, and HR. Under the HR schema, a folder named 'Tables (Filtered)' is expanded, revealing the EMPLOYEES table. A red arrow points to the EMPLOYEES table entry. To the right of the tree view is a detailed view of the EMPLOYEES table structure, showing 11 columns with their respective data types.

| COLUMN_NAME      | DATA_TYPE         |
|------------------|-------------------|
| 1 EMPLOYEE_ID    | NUMBER(6,0)       |
| 2 FIRST_NAME     | VARCHAR2(20 BYTE) |
| 3 LAST_NAME      | VARCHAR2(25 BYTE) |
| 4 EMAIL          | VARCHAR2(25 BYTE) |
| 5 PHONE_NUMBER   | VARCHAR2(20 BYTE) |
| 6 HIRE_DATE      | DATE              |
| 7 JOB_ID         | VARCHAR2(10 BYTE) |
| 8 SALARY         | NUMBER(8,2)       |
| 9 COMMISSION_PCT | NUMBER(2,2)       |
| 10 MANAGER_ID    | NUMBER(6,0)       |
| 11 DEPARTMENT_ID | NUMBER(4,0)       |

8. In the **EMPLOYEES** tab on the right, click the **Grants** subtab.

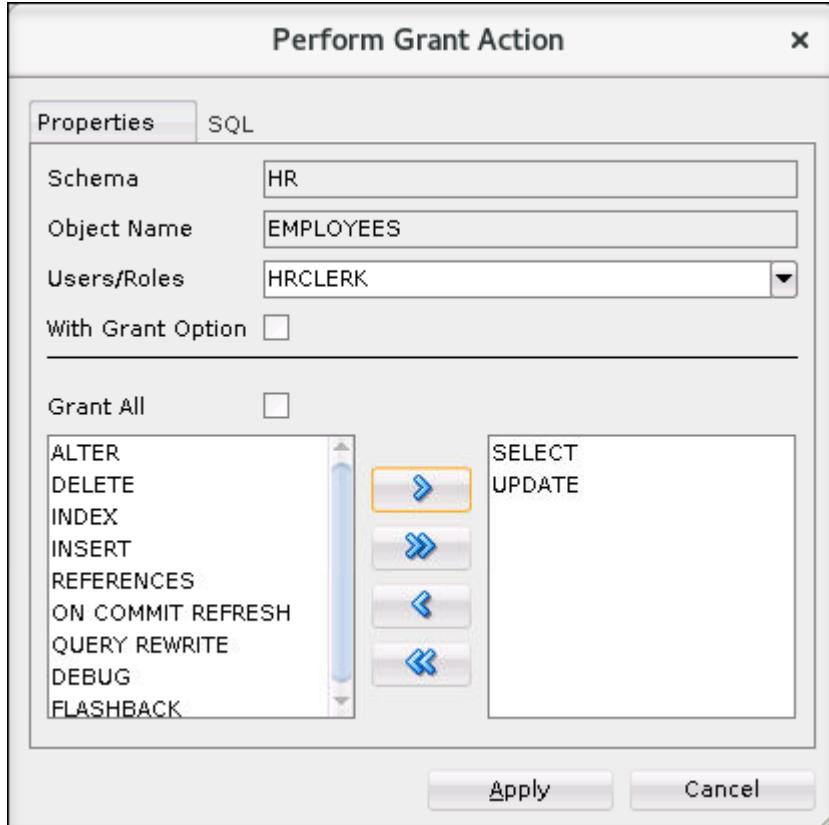
The screenshot shows the 'EMPLOYEES' tab in the Oracle SQL Developer interface, specifically focusing on the 'Grants' subtab. The 'Grants' tab is highlighted with a red box. Below it, a table displays two grants: one for 'SELECT' privilege on the 'EMPLOYEES' table and another for 'REFERENCES' privilege on the same table. The 'GRANTEE' column shows 'OE' for both grants, and the 'GRANTOR' column shows 'HR'.

| PRIVILEGE    | GRANTEE | GRANTABLE | GRANTOR | OBJECT_NAME |
|--------------|---------|-----------|---------|-------------|
| 1 SELECT     | OE      | NO        | HR      | EMPLOYEES   |
| 2 REFERENCES | OE      | NO        | HR      | EMPLOYEES   |

9. Click **Actions > Privileges > Grant**.

- The Perform Grant Action box is displayed.
- In the Users/Roles drop-down list, select **HRCLERK**.
- In the left hand list, click **SELECT** and move it to right hand list.

- d. In the left hand list, click **UPDATE** and move it to right hand list.



- e. Click the SQL tab to view the SQL statement.

```
grant UPDATE, SELECT on "HR"."EMPLOYEES" to "HRCLERK" ;
```

- f. Click **Apply**.

- g. In the Successful box click **OK**.

- h. Refresh on the Grants subtab and verify SELECT and UPDATE on HR.EMPLOYEES have been granted to HRCLERK.

| EMPLOYEES    |         |           |         |             |
|--------------|---------|-----------|---------|-------------|
| Actions...   |         |           |         |             |
| PRIVILEGE    | GRANTEE | GRANTABLE | GRANTOR | OBJECT_NAME |
| 1 SELECT     | OE      | NO        | HR      | EMPLOYEES   |
| 2 SELECT     | HRCLERK | NO        | HR      | EMPLOYEES   |
| 3 UPDATE     | HRCLERK | NO        | HR      | EMPLOYEES   |
| 4 REFERENCES | OE      | NO        | HR      | EMPLOYEES   |

### Create the HRMANAGER Role

The steps in this section are similar to the ones in the previous section.

1. Expand the PDB1–pdbadmin connection in the DBA box.
2. Expand **Security** and then select **Roles**.
3. In the Roles tab, select **Actions > Create New ...**

4. In the Create Role box,
  - a. On the Roles tab, enter Role Name: **HRMANAGER**.
  - b. Click the SQL tab to view the SQL statement.
  - c. Click **Apply**.
  - d. In the Successful box, click **OK**.
  - e. In the Create Role box, click **Close**.
5. Verify that the **HRMANAGER** role is listed in the table.

The screenshot shows the Oracle Database Roles window. The table has two columns: 'Role' and 'Authentication'. The 'Role' column lists several roles with their corresponding IDs: 49 GSM\_POOLADMIN\_ROLE, 50 HRCLERK, 51 HRMANAGER, 52 HS\_ADMIN\_EXECUTE\_ROLE, 53 HS\_ADMIN\_ROLE, and 54 HS\_ADMIN\_SELECT\_ROLE. The row for role 51 HRMANAGER is highlighted with a red border. The 'Authentication' column contains the value 'NO' for all roles.

| Role                     | Authentication |
|--------------------------|----------------|
| 49 GSM_POOLADMIN_ROLE    | NO             |
| 50 HRCLERK               | NO             |
| 51 HRMANAGER             | NO             |
| 52 HS_ADMIN_EXECUTE_ROLE | NO             |
| 53 HS_ADMIN_ROLE         | NO             |
| 54 HS_ADMIN_SELECT_ROLE  | NO             |

#### Add Object Privileges to HRMANAGER role

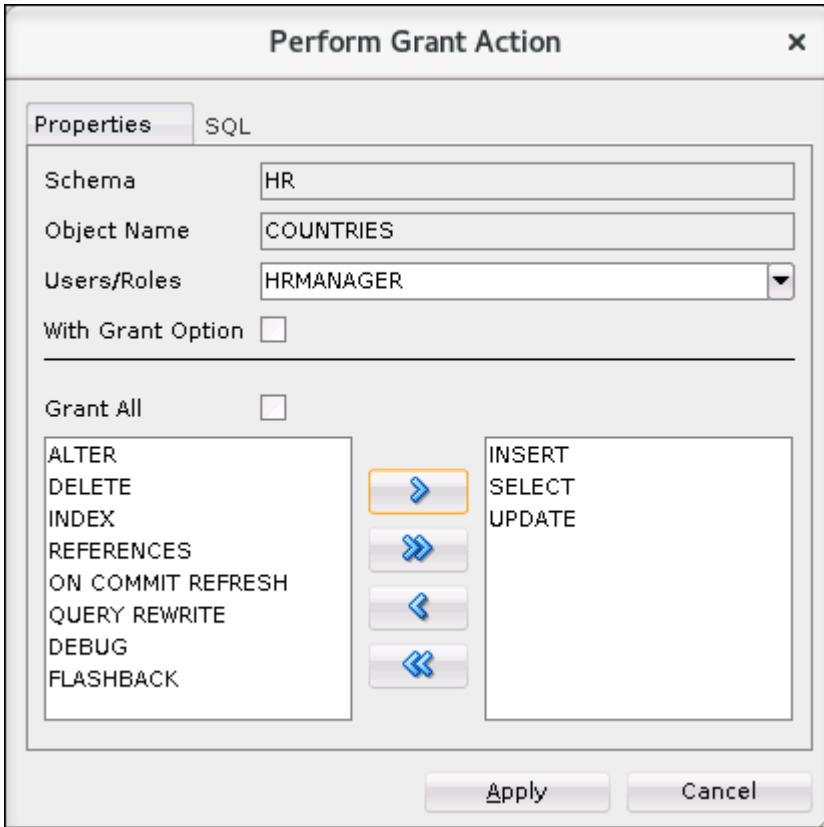
6. In the Connections box, expand **Other USERS> HR> Tables**, and click **COUNTRIES**.
7. Click the **Grants** subtab.

The screenshot shows the Oracle Database Connections window. The left pane displays a tree structure of connections and schemas. The 'HR' schema is expanded, showing tables like COUNTRIES, DEPARTMENTS, EMPLOYEES, etc. The 'COUNTRIES' table is selected and highlighted with a red border. The right pane shows the 'COUNTRIES' table details with the 'Grants' tab selected. The grants table has columns: PRIVILEGE, GRANTEE, GRANTABLE, GRANTOR, and OBJECT\_NAME. It contains two entries: 1 SELECT OE NO HR COUNTRIES and 2 REFERENCES OE NO HR COUNTRIES.

| PRIVILEGE    | GRANTEE | GRANTABLE | GRANTOR | OBJECT_NAME |
|--------------|---------|-----------|---------|-------------|
| 1 SELECT     | OE      | NO        | HR      | COUNTRIES   |
| 2 REFERENCES | OE      | NO        | HR      | COUNTRIES   |

8. Click **Actions > Privileges > Grant**.
  - a. The Perform Grant Action box is displayed.
  - b. In the Users/Roles drop-down list, select **HRMANAGER**.
  - c. In the left hand list, click **DELETE** and move it to right hand list.

- d. Repeat for **INSERT**, **SELECT**, and **UPDATE**.



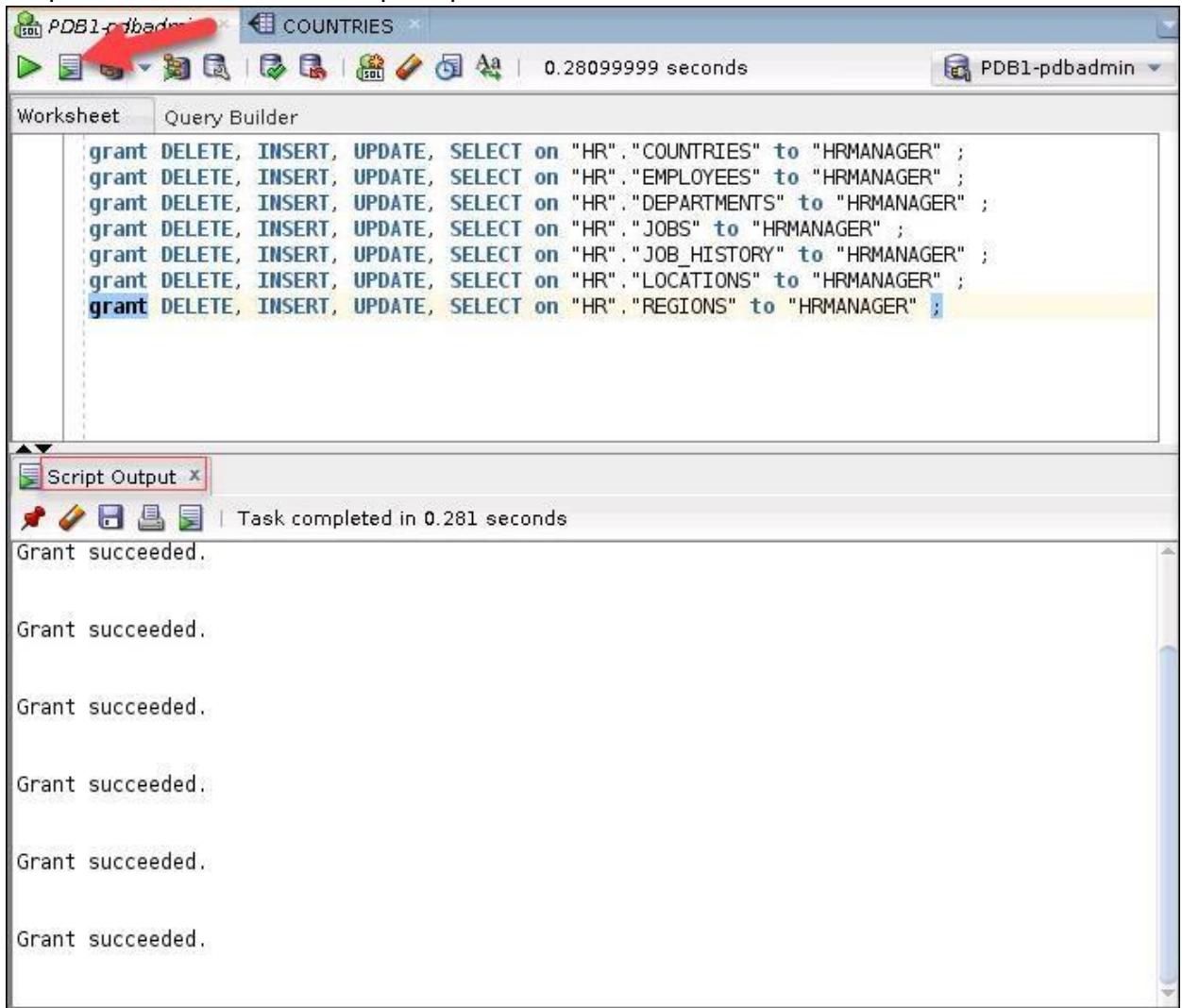
- e. Click the **SQL** tab to view the SQL statement.

```
grant DELETE, INSERT, UPDATE, SELECT on "HR"."COUNTRIES" to
"HRMANAGER" ;
```

- f. Highlight the SQL statement, and copy it (use the middle mouse button or ctrl-V).
- g. Cancel the Perform Grant Action box, click **Cancel**.
- h. Click the **PDB1-pdbadmin** tab for the Worksheet. Note if you closed the tab, you can launch a new Worksheet using ALT-F10 or select from the SQL Developer menu **Tools > SQLWorksheet**
- i. Paste the SQL Statement into the worksheet (use the middle mouse button or ctrl-V).
- j. Press **Ctrl-Enter** or the Green right arrow ➤ to execute the statement.
- k. Change **COUNTRIES** in the statement to **EMPLOYEES**, and press the Green right arrow in the menu to execute. Note: since the table name is in quotes, upper case is required.

```
grant DELETE, INSERT, UPDATE, SELECT on "HR"."EMPLOYEES" to
"HRMANAGER" ;
```

- I. Repeat changing the table name to **EMPLOYEES**, **DEPARTMENTS**, **JOBS**, **JOB\_HISTORY**, **LOCATIONS**, **REGIONS**. Execute all the statements by clicking run script  button. Note the script output window:



The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a red arrow pointing to the 'Run Script' button (a green triangle with a white circle). The main area is a 'Worksheet' tab showing a block of SQL grants:

```
grant DELETE, INSERT, UPDATE, SELECT on "HR"."COUNTRIES" to "HRMANAGER" ;
grant DELETE, INSERT, UPDATE, SELECT on "HR"."EMPLOYEES" to "HRMANAGER" ;
grant DELETE, INSERT, UPDATE, SELECT on "HR"."DEPARTMENTS" to "HRMANAGER" ;
grant DELETE, INSERT, UPDATE, SELECT on "HR"."JOBS" to "HRMANAGER" ;
grant DELETE, INSERT, UPDATE, SELECT on "HR"."JOB_HISTORY" to "HRMANAGER" ;
grant DELETE, INSERT, UPDATE, SELECT on "HR"."LOCATIONS" to "HRMANAGER" ;
grant DELETE, INSERT, UPDATE, SELECT on "HR"."REGIONS" to "HRMANAGER" ;
```

Below the worksheet is a 'Script Output' tab showing the results of the grants:

```
Task completed in 0.281 seconds
Grant succeeded.
Grant succeeded.
Grant succeeded.
Grant succeeded.
Grant succeeded.
```

9. Verify the **HRMANAGER** role and required privileges.
- In the DBA box, expand **Security** and click **Roles**.
  - Double-click the **HRMANAGER** role in the view on the right.
  - In the **HRMANAGER** tab, click the **Object Privs** subtab.

- d. Verify the privileges granted on the HR tables are shown.

The screenshot shows the SQL\*Developer interface with the 'Object Privils' tab selected. The table displays 24 grants for the 'HRMANAGER' role across four HR tables: COUNTRIES, DEPARTMENTS, EMPLOYEES, and LOCATIONS. The grants include various combinations of DELETE, INSERT, SELECT, and UPDATE operations.

|    | Object Privilege | Schema | Object      |
|----|------------------|--------|-------------|
| 1  | DELETE           | HR     | COUNTRIES   |
| 2  | INSERT           | HR     | COUNTRIES   |
| 3  | SELECT           | HR     | COUNTRIES   |
| 4  | UPDATE           | HR     | COUNTRIES   |
| 5  | DELETE           | HR     | DEPARTMENTS |
| 6  | INSERT           | HR     | DEPARTMENTS |
| 7  | SELECT           | HR     | DEPARTMENTS |
| 8  | UPDATE           | HR     | DEPARTMENTS |
| 9  | DELETE           | HR     | EMPLOYEES   |
| 10 | INSERT           | HR     | EMPLOYEES   |
| 11 | SELECT           | HR     | EMPLOYEES   |
| 12 | UPDATE           | HR     | EMPLOYEES   |
| 13 | DELETE           | HR     | JOB_HISTORY |
| 14 | INSERT           | HR     | JOB_HISTORY |
| 15 | SELECT           | HR     | JOB_HISTORY |
| 16 | UPDATE           | HR     | JOB_HISTORY |
| 17 | DELETE           | HR     | LOCATIONS   |
| 18 | INSERT           | HR     | LOCATIONS   |
| 19 | SELECT           | HR     | LOCATIONS   |
| 20 | UPDATE           | HR     | LOCATIONS   |
| 21 | DELETE           | HR     | REGIONS     |
| 22 | INSERT           | HR     | REGIONS     |
| 23 | SELECT           | HR     | REGIONS     |
| 24 | UPDATE           | HR     | REGIONS     |

10. Exit SQL\*Developer by clicking **File > Exit**

# **Practices for Lesson 21: Configuring User Resource Limits**



## **Practices for Lesson 21: Overview**

---

### **Overview**

In these practices, you will create and manage user profiles.

## Practice 21-1: Using SQL\*Developer to Create a Local Profile

---

### Overview

In this practice, the `PDBADMIN` user (local administrator for `ORCLPDB1`) 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.

### Assumptions

You are currently logged in as the `oracle` OS user.

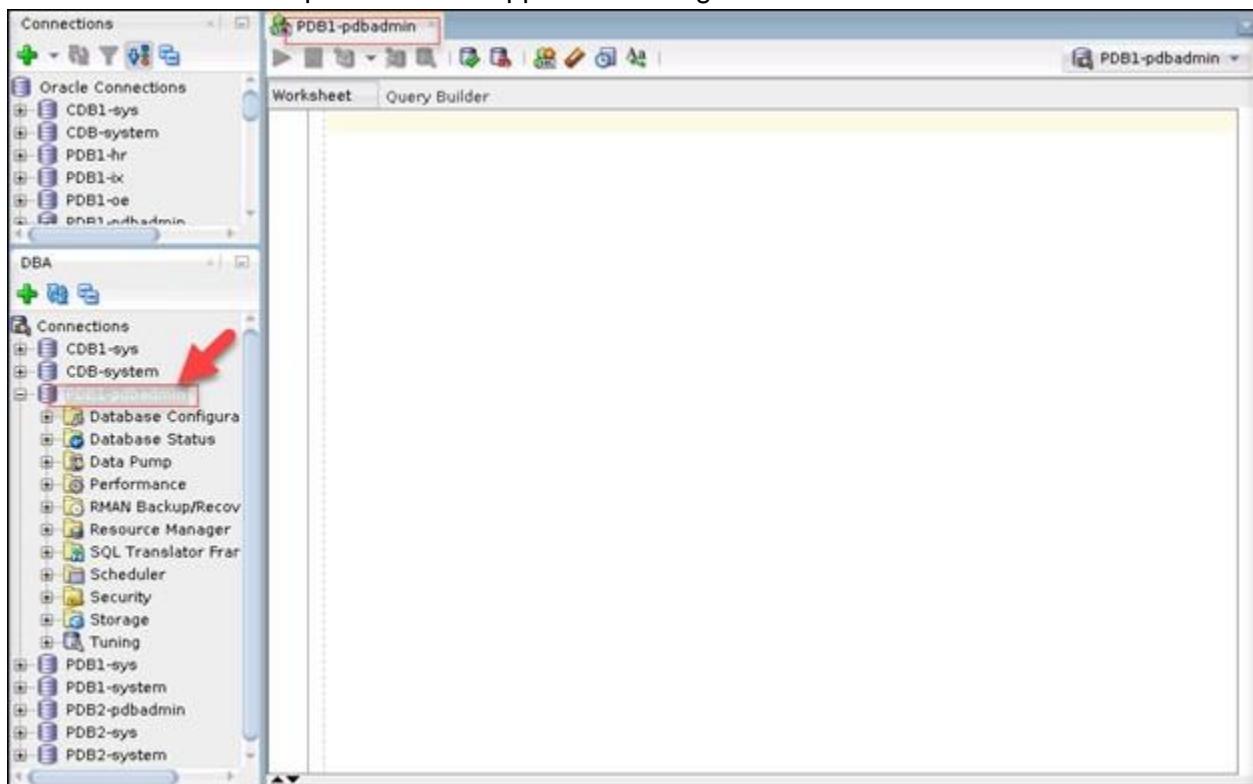
The `PDBADMIN` user has been granted the `DBA` role.

### Tasks

#### Log in to SQL\*Developer (ORCLPDB1)

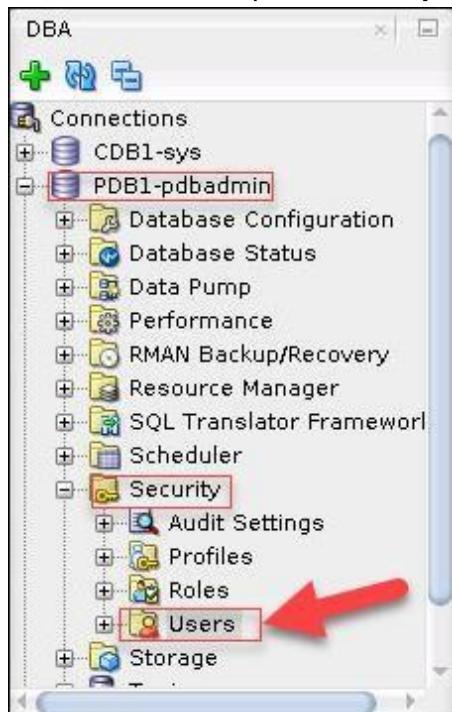
1. Launch SQL\*Developer.
2. Click Menu View > DBA
  - a. In the DBA box, double click: `PDB1-pdbadmin`.

- b. A worksheet for PDB1-pdbadmin will appear on the right.



#### View Privileges and Roles for PDBADMIN

3. In the DBA box, expand **PDB1-pdadmin > Security> Users**



4. On the Users tab, select PDBADMIN

A screenshot of the Oracle SQL Developer interface. The title bar shows 'PDB1-pdbadmin' and the tab is 'Users'. The main area is a table with columns: User Name, Account Status, Expiration Date, Default Tablespace, and Temporary Tablespace. The table contains 15 rows, each representing a database user. A red arrow points to the row for '31 PDBADMIN', which is highlighted with a red border. The data in the table is as follows:

| User Name                 | Account Status | Expiration Date | Default Tablespace | Temporary Tablespace |
|---------------------------|----------------|-----------------|--------------------|----------------------|
| 18 IX                     | OPEN           | (null)          | USERS              | TEMP                 |
| 19 LBACSYS                | LOCKED         | (null)          | SYSTEM             | TEMP                 |
| 20 MDDATA                 | LOCKED         | (null)          | USERS              | TEMP                 |
| 21 MDSYS                  | LOCKED         | (null)          | SYSAUX             | TEMP                 |
| 22 OE                     | OPEN           | (null)          | USERS              | TEMP                 |
| 23 OJVMSYS                | LOCKED         | (null)          | USERS              | TEMP                 |
| 24 OLAPSYS                | LOCKED         | (null)          | SYSAUX             | TEMP                 |
| 25 ORACLE_OCM             | LOCKED         | (null)          | USERS              | TEMP                 |
| 26 ORDDATA                | LOCKED         | (null)          | USERS              | TEMP                 |
| 27 ORDPLUGINS             | LOCKED         | (null)          | USERS              | TEMP                 |
| 28 ORDSYS                 | LOCKED         | (null)          | USERS              | TEMP                 |
| 29 OUTLN                  | LOCKED         | (null)          | SYSTEM             | TEMP                 |
| 30 PDB1_ADMIN             | OPEN           | (null)          | USERS              | TEMP                 |
| 31 PDBADMIN               | OPEN           | (null)          | USERS              | TEMP                 |
| 32 PM                     | OPEN           | (null)          | USERS              | TEMP                 |
| 33 REMOTE_SCHEDULER_AGENT | LOCKED         | (null)          | USERS              | TEMP                 |

5. The PDBADMIN tab is displayed. Click the Roles subtab. Notice that DBA is a ROLE.

A screenshot of the Oracle SQL Developer interface. The title bar shows 'PDB1-pdbadmin' and the tab is 'PDBADMIN'. Below it, a subtab bar has 'Roles' selected, indicated by a red border. The main area is a table with columns: Role, Admin Option, and Default. The table contains 4 rows, each representing a role assigned to the PDBADMIN user. The data in the table is as follows:

| Role        | Admin Option | Default |
|-------------|--------------|---------|
| 1 DBA       | NO           | YES     |
| 2 HRCLERK   | YES          | YES     |
| 3 HRMANAGER | YES          | YES     |
| 4 PDB_DBA   | YES          | YES     |

6. In the DBA box, expand **Security** and **Roles** and select **DBA**.

The screenshot shows the Oracle Database Control interface. On the left, the 'Connections' sidebar lists 'Oracle Connections' and 'CDB1-sys'. Under 'DBA', the 'Connections' section is expanded, showing 'CDB1-sys', 'PDB1-pdbadmin', and its sub-nodes: 'Database Configuration', 'Database Status', 'Data Pump', 'Performance', 'RMAN Backup/Recovery', 'Resource Manager', 'SQL Translator Framework', 'Scheduler', 'Security', 'Audit Settings', 'Profiles', 'Roles' (which is highlighted with a red box), and 'Users'. 'Storage' is also listed. The main pane is titled 'PDB1-pdbadmin Roles' and shows a list of roles from 13 to 31. Role 17, 'DBA', is highlighted with a red arrow.

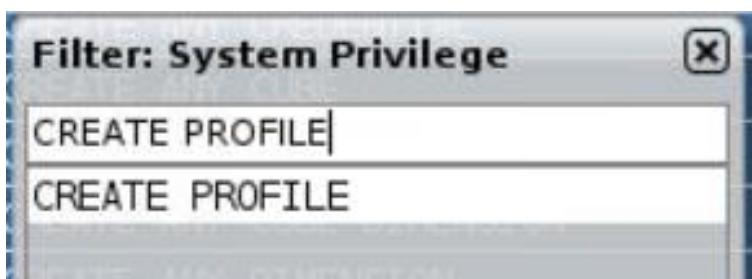
| Role                          | Authentication |
|-------------------------------|----------------|
| 13 CTXAPP                     | NO             |
| 14 DATAPATCH_ROLE             | NO             |
| 15 DATAPUMP_EXP_FULL_DATABASE | NO             |
| 16 DATAPUMP_IMP_FULL_DATABASE | NO             |
| 17 DBA                        | NO             |
| 18 DBFS_ROLE                  | NO             |
| 19 DBJAVASCRIPT               | NO             |
| 20 DBMS_MDX_INTERNAL          | NO             |
| 21 DV_ACCTMGR                 | NO             |
| 22 DV_ADMIN                   | NO             |
| 23 DV_AUDIT_CLEANUP           | NO             |
| 24 DV_DATAPUMP_NETWORK_LINK   | NO             |
| 25 DV_GOLDENGATE_ADMIN        | NO             |
| 26 DV_GOLDENGATE_REDO_ACCESS  | NO             |
| 27 DV_MONITOR                 | NO             |
| 28 DV_OWNER                   | NO             |
| 29 DV_PATCH_ADMIN             | NO             |
| 30 DV_POLICY_OWNER            | NO             |
| 31 DV_PUBLIC                  | NO             |

7. In the DBA tab, click the **Sys Prvcs** subtab. Scroll down through the list of privileges.

The screenshot shows the DBA tab with the 'Sys Prvcs' subtab selected (highlighted with a red box). The subtab list includes 'General', 'Roles', 'Sys Prvcs', 'Object Prvcs', 'Consumer Group Prvcs', and 'Us'. Below the subtab list is a table titled 'System Privilege' with columns 'System Privilege' and 'Admin Option'. The table lists seven system privileges: 1 ADMINISTER ANY SQL TUNING SET, 2 ADMINISTER DATABASE TRIGGER, 3 ADMINISTER RESOURCE MANAGER, 4 ADMINISTER SQL MANAGEMENT OBJECT, 5 ADMINISTER SQL TUNING SET, 6 ADVISOR, and 7 ALTER ANY ANALYTIC VIEW.

| System Privilege                   | Admin Option |
|------------------------------------|--------------|
| 1 ADMINISTER ANY SQL TUNING SET    | NO           |
| 2 ADMINISTER DATABASE TRIGGER      | NO           |
| 3 ADMINISTER RESOURCE MANAGER      | NO           |
| 4 ADMINISTER SQL MANAGEMENT OBJECT | NO           |
| 5 ADMINISTER SQL TUNING SET        | NO           |
| 6 ADVISOR                          | NO           |
| 7 ALTER ANY ANALYTIC VIEW          | NO           |

8. Hover over the System Privilege header. Click the Filter icon. To look for specific privileges, for example, **CREATE PROFILE**, type **CREATE PROFILE** in the **Filter: System Privilege** field and press **Enter**.



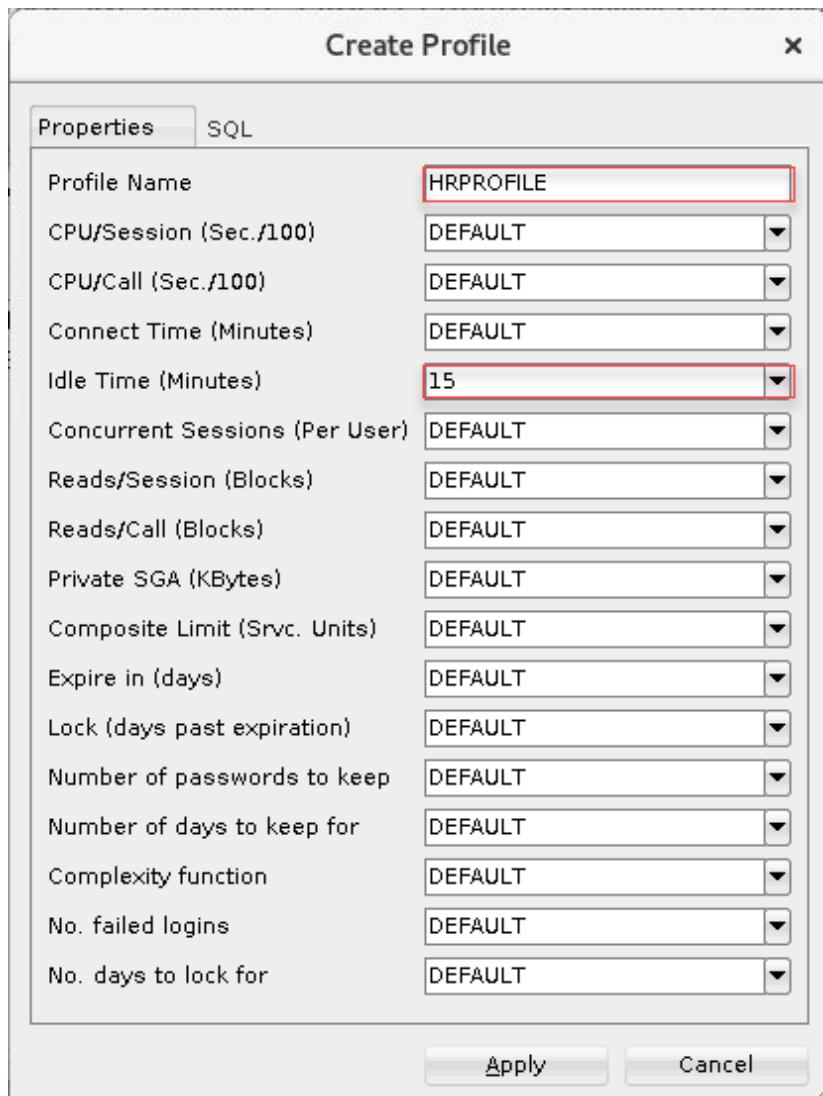
9. Close the DBA Role tab.

#### Create a Local Profile

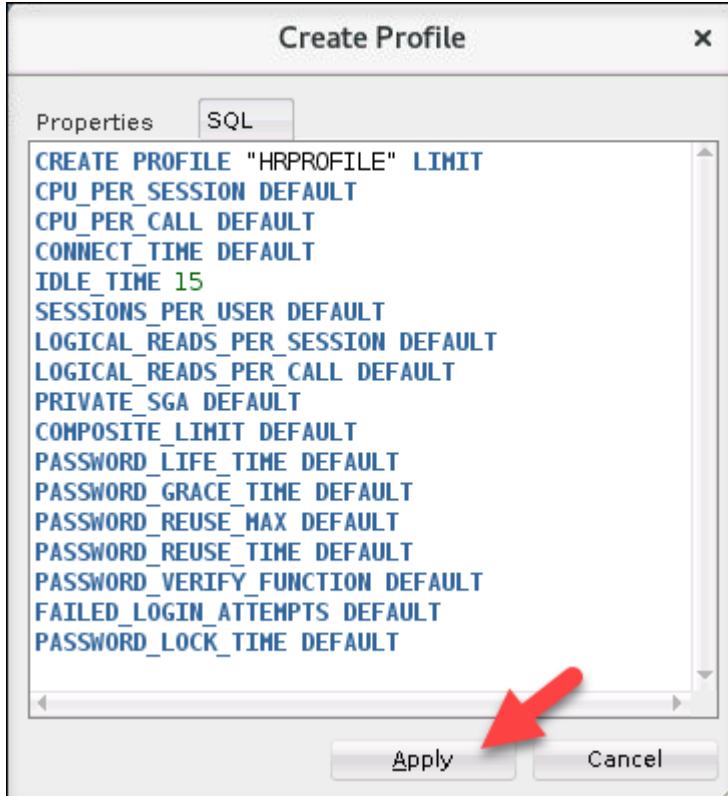


10. In the DBA box, expand **Security** and click **Profiles**.
11. In the Profiles tab, click **Actions > Create New ...**

12. In the Create Profile box, enter:
- Profile Name: **HRPROFILE**
  - IdleTime (minutes): **15**
- Leave all other fields set to Default.



13. Click the **SQL** tab to review the SQL command for this task. When done, click **APPLY**.



14. In the Successful window, click **OK**.

15. Verify that **HRPROFILE** is in the list of profiles. Note, you may need to click **Refresh**

#### **Set the RESOURCE\_LIMIT Initialization Parameter**

16. In the DBA box, expand **Database Configuration** and select **Initialization Parameters**.

17. Hoover over the Parameter header and click the **Filter** icon. Type **resource\_limit** in the Filter: Parameter field and press **Enter**.



18. The **RESOURCE\_LIMIT** initialization parameter is listed in the table. Verify that the **RESOURCE LIMIT** value is set to **TRUE**.
19. If **RESOURCE\_LIMIT** is not set to **true**, perform the following steps:
- Double-click the Value field of the **RESOURCE\_LIMIT** row.
  - Select **TRUE**, and click somewhere outside the Value field.
  - The row number will have an asterisk, then you can press F11 or click the icon to commit the change,
  - In the Commit Strategy dialog box, check both **Memory** and **SPFILE** boxes and then click **Apply**.
20. Close the SQL\*Developer 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.

21. Open a new terminal window and source the **oraenv** script for the **orclcldb** database. .

```
$. oraenv
ORACLE_SID = [orclcldb] ? orclcldb
```

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

22. Connect to ORCLPDB1 as the local DBA, PDBADMIN. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$ sqlplus pdbadmin/password@orclpdb1
...
SQL>
```

23. 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>
```

24. 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' order by 1,2;

RESOURCE RESOURCE_NAME LIMIT

KERNEL COMPOSITE_LIMIT DEFAULT
KERNEL CONNECT_TIME DEFAULT
KERNEL CPU_PER_CALL DEFAULT
KERNEL CPU_PER_SESSION DEFAULT
KERNEL IDLE_TIME 15
KERNEL LOGICAL_READS_PER_CALL DEFAULT
KERNEL LOGICAL_READS_PER_SESSION DEFAULT
KERNEL PRIVATE_SGA DEFAULT
KERNEL SESSIONS_PER_USER DEFAULT
PASSWORD FAILED_LOGIN_ATTEMPTS DEFAULT
PASSWORD INACTIVE_ACCOUNT_TIME DEFAULT
PASSWORD PASSWORD_GRACE_TIME DEFAULT
PASSWORD PASSWORD_LIFE_TIME DEFAULT
PASSWORD PASSWORD_LOCK_TIME DEFAULT
PASSWORD PASSWORD_REUSE_MAX DEFAULT
PASSWORD PASSWORD_REUSE_TIME DEFAULT
PASSWORD PASSWORD_VERIFY_FUNCTION 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.

25. Investigate by displaying the full error message that you received in step 4. 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>
```

26. Set an appropriate limit. Because 10 is too low, use the lowest valid number instead, which would be 15.

```
SQL> alter profile hrprofile limit inactive_account_time 15;

Profile altered.

SQL>
```

27. 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' order by 1,2;

RESOURCE RESOURCE_NAME LIMIT

KERNEL COMPOSITE_LIMIT DEFAULT
KERNEL CONNECT_TIME DEFAULT
KERNEL CPU_PER_CALL DEFAULT
KERNEL CPU_PER_SESSION DEFAULT
KERNEL IDLE_TIME 15
KERNEL LOGICAL_READS_PER_CALL DEFAULT
```

```

KERNEL LOGICAL_READS_PER_SESSION DEFAULT
KERNEL PRIVATE_SGA DEFAULT
KERNEL SESSIONS_PER_USER DEFAULT
PASSWORD FAILED_LOGIN_ATTEMPTS DEFAULT
PASSWORD INACTIVE_ACCOUNT_TIME 15
PASSWORD PASSWORD_GRACE_TIME DEFAULT
PASSWORD PASSWORD_LIFE_TIME DEFAULT
PASSWORD PASSWORD_LOCK_TIME DEFAULT
PASSWORD PASSWORD_REUSE_MAX DEFAULT
PASSWORD PASSWORD_REUSE_TIME DEFAULT
PASSWORD PASSWORD_VERIFY_FUNCTION DEFAULT

17 rows selected.

SQL>

```

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

```

SQL> exit
...
$ exit

```

29. 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 21-2: Using SQL\*Developer to Create Local Users

---

### Overview

In this practice you use SQL\*Developer to connect to ORCLPDB1 as the PDBADMIN user and create local user accounts according to the following table. Assign the profile named HRP PROFILE 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 HRP PROFILE.

| Name          | User Account | Description      | Privileges/Roles                                           | Method to Use to Create User           |
|---------------|--------------|------------------|------------------------------------------------------------|----------------------------------------|
| Jenny Goodman | JGOODMAN     | A new HR manager | CREATE SESSION privilege<br>HRCLERK role<br>HRMANAGER role | SQL*Developer                          |
| David Hamby   | DHAMBY       | A new HR clerk   | CREATE SESSION privilege<br>HRCLERK role                   | SQL*Developer                          |
| Rachel Pandya | RPANDYA      | A new HR clerk   | CREATE SESSION privilege<br>HRCLERK role                   | SQL script with substitution variables |

### Assumptions

You are currently logged in as the oracle user.

You created a local profile in ORCLPDB1 named HRP PROFILE and two local roles (HRCLERK and HRMANAGER) in ORCLPDB1. You also assigned the DBA role to the PDBADMIN user, which is the local administrator for ORCLPDB1. If you haven't done so, complete the following practices before starting this one:

- Practice 2-1 Granting the DBA Role to PDBADMIN
- Practice 2-2 Creating Local Roles Using SQL\*Developer
- Practice 3-1 Creating a Local Profile Using SQL\*Developer

### Tasks

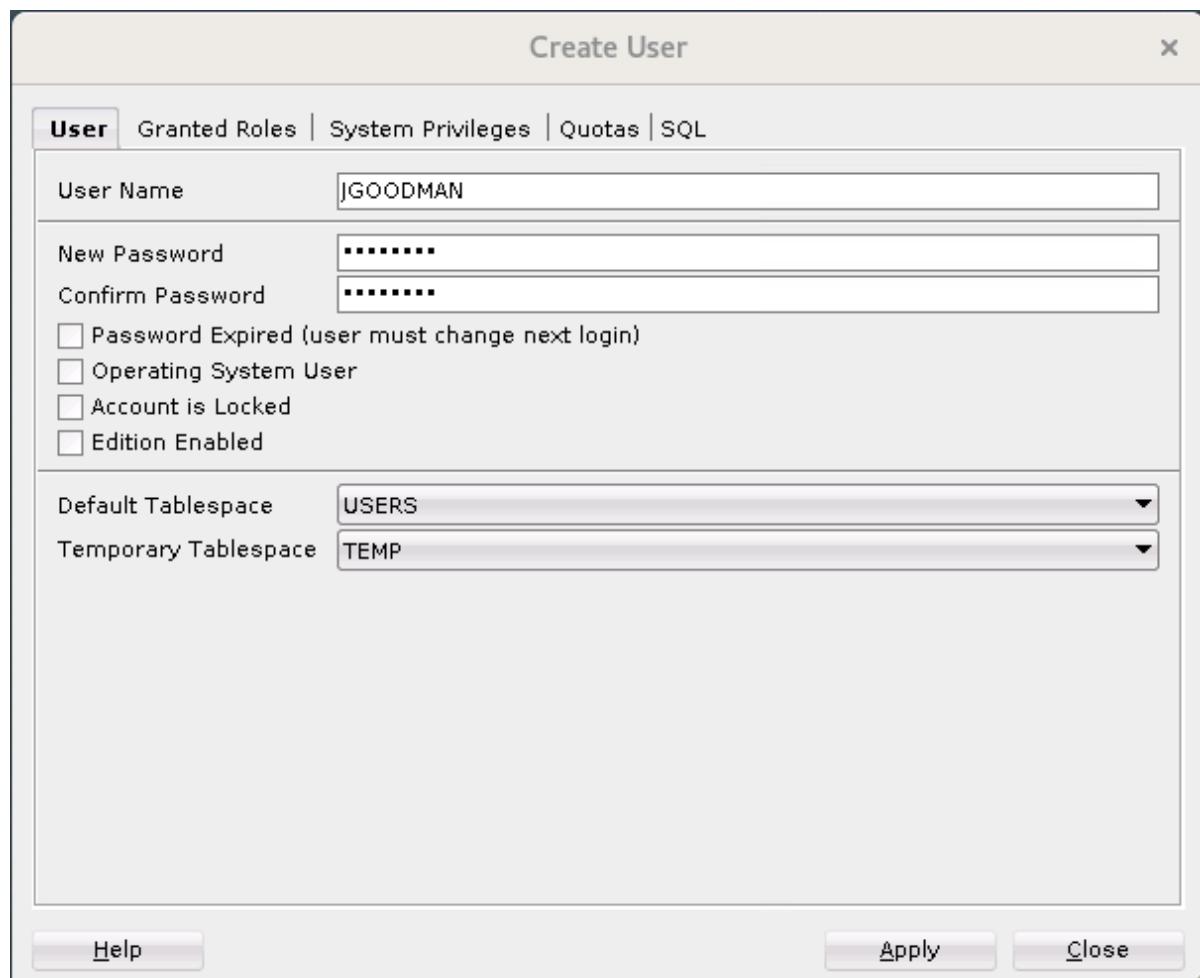
#### Connect to ORCLPDB1 as PDBADMIN Using SQL\*Developer

1. Launch SQL\*Developer from your Desktop. In the Connections box, double-click **PDB1-pdbadmin**.

## Create a User Account for Jenny Goodman

In this section, you create a user account named `JGOODMAN` by using the SQL\*Developer interface.

2. In the DBA box, expand **PDB1-pdbadmin**.
3. Expand **Security** and select **Users**.
4. Click **Actions > Create New ...**.
5. The Create User dialog is displayed. On the User tab, enter or select the following values.
  - Name: Enter `JGOODMAN`
  - Enter the **password** and confirm. Refer to *Course Practice Environment: Security Credentials* for the password value.
  - Leave the check boxes for Password Expired, Operating System User, Account is Locked, and Edition Enabled deselected.
  - Default Tablespace: `USERS`
  - Temporary Tablespace: `TEMP`



6. On the Granted Roles tab, check the Granted box for **HRCLERK** role, and **HRMANAGER** role.

| Role Name             | Granted                             | Admin                    | Default                  |
|-----------------------|-------------------------------------|--------------------------|--------------------------|
| HRCLERK               | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| HRMANAGER             | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| HS_ADMIN_EXECUTE_ROLE | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> |
| HS_ADMIN_ROLE         | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> |

7. On the System Privileges tab, check the Granted box for **Create Session** privilege.

| Privilege                      | Granted                             | Admin Option             |
|--------------------------------|-------------------------------------|--------------------------|
| CREATE SEQUENCE                | <input type="checkbox"/>            | <input type="checkbox"/> |
| CREATE SESSION                 | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| CREATE SQL TRANSLATION PROFILE | <input type="checkbox"/>            | <input type="checkbox"/> |
| CREATE SYNONYM                 | <input type="checkbox"/>            | <input type="checkbox"/> |
| CREATE TABLE                   | <input type="checkbox"/>            | <input type="checkbox"/> |

8. On the SQL tab, review the SQL statement, click **Apply**.

Note: The **password** value will be the one from *Course Practice Environment: Security Credentials*.

```
-- USER SQL
CREATE USER "JGOODMAN" IDENTIFIED BY "password"
DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP";

-- QUOTAS

-- ROLES
GRANT "HRMANAGER" TO "JGOODMAN" ;
GRANT "HRCLERK" TO "JGOODMAN" ;

-- SYSTEM PRIVILEGES
GRANT CREATE SESSION TO "JGOODMAN" ;
```

**Note:** In Practice 3-3 Configuring a Default Role for a User, you will assign the **HRCLERK** role to be this user account's default role.

9. In the Successful dialog box, click **OK**.

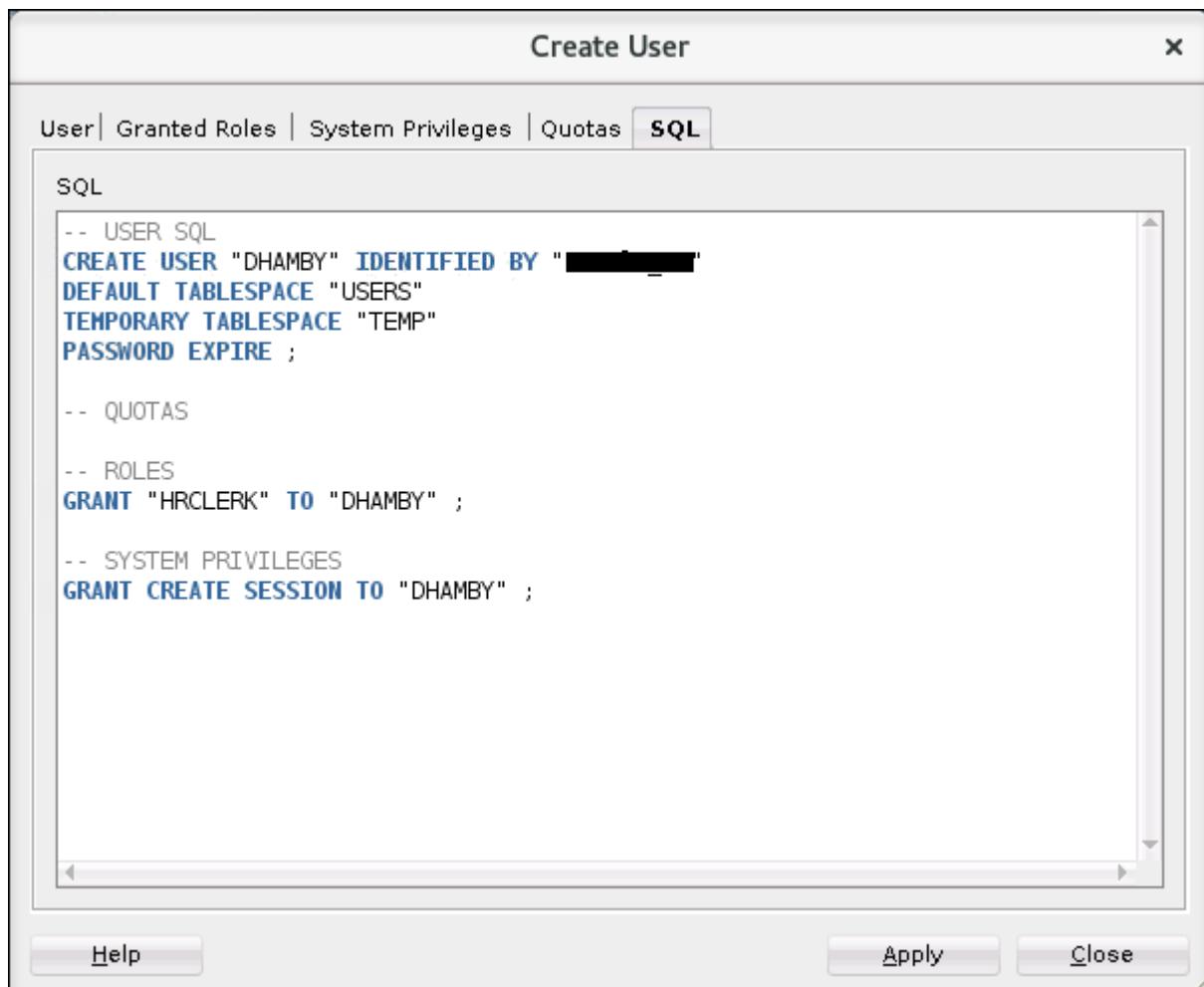
10. Verify that **JGOODMAN** is listed in the Users table. From here, you can see that the **JGOODMAN** account is unlocked and the password has not expired.

## Create a User Account for David Hamby

In this section, you create another user account named **DHAMBY** by using the SQL\*Developer 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.

11. If the Users tab is not open, expand **Security** and then select **Users**.
12. Click **Actions > Create New ...**
13. The Create User dialog is displayed. On the User Account page, enter or select the following values.
  - Name: Enter **DHAMBY**
  - Set the *password* and confirm. See *Course Practice Environment: Security Credentials* for password values.
  - Select the Password Expired check box to force the user to change his password at logon.
  - Default Tablespace: **USERS**
  - Temporary Tablespace: **TEMP**
14. On the Granted Roles tab, select **HRCLERK** role and check **Granted**.
15. On the System Privileges tab, select the **CREATE SESSION** privilege, and check **Granted**.

16. Click **SQL** tab.



17. 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. This script will be used 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 the SQL statements in the previous step to the clipboard.
- Click **Apply** to create the **DHAMBY** user.
- Click **OK** in the Successful box.
- Click the **PDB1-pdadmin** tab to view the worksheet.

- e. **Paste** the SQL statements in the worksheet.

Change every occurrence as follows:

| Item To Change | Change To  |
|----------------|------------|
| DHAMBY         | &&username |
| oracle_4U      | &&password |
| HRCLERK        | &&role     |

- f. Verify that the code looks like the following code. Don't worry if your GRANT statements are in a different order. See *Course Practice Environment: Security Credentials* for password values.

```
-- USER SQL
CREATE USER "&&username" IDENTIFIED BY "&&password"
DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP"
PASSWORD EXPIRE ;

-- QUOTAS

-- ROLES
GRANT "&&role" TO "&&username" ;

-- SYSTEM PRIVILEGES
GRANT CREATE SESSION TO "&&username" ;
```

- g. Save this script. Click the Save file Icon or **ctrl-S**.

- h. In the Save dialog, Browse to **/home/oracle/labs**. In the File Name box, enter **CreateHRUser.sql**. Click **Save**. The file is saved and formatted.

18. Verify that **DHAMBY** is listed in the Users table. The account is unlocked and the password has expired.

| User Name   | Account Status | Expiration Date | Default Tablespace | Temporary Tablespace | Profile | Created         | User Type | Shard User |
|-------------|----------------|-----------------|--------------------|----------------------|---------|-----------------|-----------|------------|
| 7 DBSPWUSER | LOCKED         | (null)          | SYSAUX             | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |
| 8 DBSNMP    | LOCKED         | (null)          | SYSAUX             | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |
| 9 DHAMBY    | EXPIRED        | 21-OCT-20       | USERS              | TEMP                 | DEFAULT | 21-OCT-20 LOCAL | NO        |            |
| 10 DTP      | LOCKED         | (null)          | USERS              | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |
| 11 DVE      | LOCKED         | (null)          | SYSAUX             | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |

19. Minimize, but don't close, SQL\*Developer.

## 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 SQL\*Developer interface to create this user, you use the SQL script that you generated in the previous section.

20. Source the oraenv script for the orclcdb database.

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

21. Start SQL\*Plus and connect to ORCLPDB1 as the PDBADMIN user. See *Course Practice Environment: Security Credentials* for **password** values.

```
$ sqlplus pdbadmin/password@orclpdb1
...
SQL>
```

22. Execute the **CreateHRUser.sql** script. Enter RPANDYA when prompted for the username. Enter the password for the user from the *Course Practice Environment: Security Credentials* when prompted for the password. Enter HRCLERK when prompted for the role. The order of the GRANT statements does not matter.

```
SQL> @/home/oracle/labs/CreateHRUser.sql
Enter value for username: RPANDYA
Enter value for password: password
old 1: CREATE USER "&&username" IDENTIFIED BY "&&password"
new 1: CREATE USER "RPANDYA" IDENTIFIED BY "fenago"

User created.

Enter value for role: HRCLERK
old 1: GRANT "&&role" TO "&&username"
new 1: GRANT "HRCLERK" TO "RPANDYA"

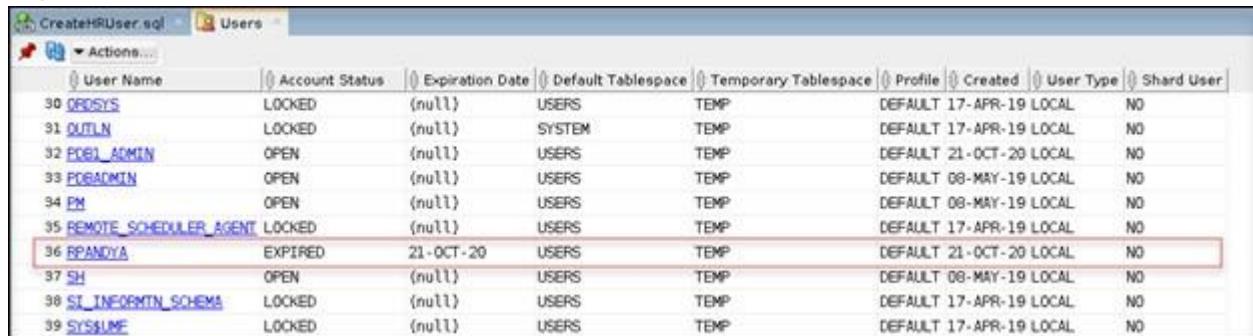
Grant succeeded.

old 1: GRANT CREATE SESSION TO "&&username"
new 1: GRANT CREATE SESSION TO "RPANDYA"

Grant succeeded.

SQL>
```

23. Return to SQL\*Developer and click the Refresh  button in the **Users** tab to refresh the data then scroll down the list and verify that the user RPANDYA has been created as expected.



| User Name                 | Account Status | Expiration Date | Default Tablespace | Temporary Tablespace | Profile | Created         | User Type | Shard User |
|---------------------------|----------------|-----------------|--------------------|----------------------|---------|-----------------|-----------|------------|
| 30 _DOSYS                 | LOCKED         | (null)          | USERS              | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |
| 31 _DUTLN                 | LOCKED         | (null)          | SYSTEM             | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |
| 32 PDB\$ADMIN             | OPEN           | (null)          | USERS              | TEMP                 | DEFAULT | 21-OCT-20 LOCAL | NO        |            |
| 33 PMADMIN                | OPEN           | (null)          | USERS              | TEMP                 | DEFAULT | 08-MAY-19 LOCAL | NO        |            |
| 34 PM                     | OPEN           | (null)          | USERS              | TEMP                 | DEFAULT | 08-MAY-19 LOCAL | NO        |            |
| 35 REMOTE_SCHEDULER_AGENT | LOCKED         | (null)          | USERS              | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |
| 36 RPANDYA                | EXPIRED        | 21-OCT-20       | USERS              | TEMP                 | DEFAULT | 21-OCT-20 LOCAL | NO        |            |
| 37 SH                     | OPEN           | (null)          | USERS              | TEMP                 | DEFAULT | 08-MAY-19 LOCAL | NO        |            |
| 38 _ST_INFORMTN_SCHEMA    | LOCKED         | (null)          | USERS              | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |
| 39 SYSUME                 | LOCKED         | (null)          | USERS              | TEMP                 | DEFAULT | 17-APR-19 LOCAL | NO        |            |

24. Close the SQL\*Developer window.

### Test DHAMBY's Access in SQL\*Plus

Connect to ORCLPDB1 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 2-2 Using SQL\*Developer to Create Local Roles*, 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.

25. Return to the terminal window.

26. Connect to ORCLPDB1 as DHAMBY. When prompted, enter the new **password**. See Course *Practice Environment: Security Credentials* for the original and new **password**. When you enter the new password, it is not displayed in the interface.

```
SQL> connect dhambay/original-password@orclpdb1
ERROR:
ORA-28001: the password has expired

Changing password for dhambay
New password: new-password
Retype new password: new-password
Password changed
Connected.

SQL>
```

27. 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>
```

28. 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>
```

29. Disconnect from ORCLPDB1.

```
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.

30. Connect to ORCLPDB1 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@orclpdb1
Connected.
SQL>
```

31. 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>
```

32. 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>
```

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

```
SQL> rollback;
```

```
Rollback complete.
```

```
SQL>
```

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

```
SQL> select count(*) from hr.employees;
```

```
COUNT (*)
```

```

```

```
107
```

```
SQL>
```

35. Disconnect from ORCLPDB1.

```
SQL> disconnect
```

```
...
```

```
SQL>
```

## Test the Idle Time Limit in HRP PROFILE

If you recall, in *Practice 3-1 Using SQL\*Developer to Create a Local Profile*, you created a profile named `HRPROFILE`. In that profile, you configured the Idle Time limit to be 15 minutes. Assign this profile to all three users (JGOODMAN, DHAMBY, and RPANDYA). In this section, test that limit by connecting to `ORCLPDB1` 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.

36. Connect to `ORCLPDB1` as `PDBADMIN` and assign the profile named `HRPROFILE` to all three users (JGOODMAN, DHAMBY, and RPANDYA). Refer to *Course Practice Environment: Security Credentials* for the `password` value

```
SQL> connect pdbadmin/password@orclpdb1
Connected.
SQL> alter user jgoodman profile hrprofile;

User altered.

SQL> alter user dhamby profile hrprofile;

User altered.

SQL> alter user rpandya profile hrprofile;

User altered.

SQL>
```

37. Connect to ORCLPDB1 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/original-password@orclpdb1
ERROR:
ORA-28001: the password has expired

Changing password for rpandya
New password: new-password
Retype new password: new-password
Password changed
Connected.
SQL>
```

38. Wait for 15 minutes. You can leave this terminal window open while waiting.
39. After 15 minutes, query the salary for employee 197 from the HR.EMPLOYEES table. The query returns the message "exceeded maximum idle time..." which indicates that HRPROFILE is working.

```
SQL> select salary from hr.employees where employee_id=197;
ERROR at line 1:

ORA-02396: exceeded maximum idle time, please connect again
SQL>
```

40. Exit SQL\*Plus

```
SQL> exit
...
$
```

41. Exit all terminals.

## Practice 21-3: 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 ORCLPDB1). Jenny logs in to ORCLPDB1 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 the HRMANAGER role to it, as well as the less-privileged HRCLERK role. To complete this practice, you must first complete the following practices:

- Practice 2-1 Granting the DBA Role to PDBADMIN
- Practice 2-2 Practice

### Tasks

#### Configure a Default Role for JGOODMAN

1. Open a new terminal, use the oraenv command to source the orclcdb database, then start SQL\*Plus and connect to ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus pdbadmin/password@orclpdb1
...
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> <b>alter user jgoodman default role hrclerk;</b> |  |
| 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> <b>select granted_role, default_role from dba_role_privs where grantee='JGOODMAN';</b> |       |
| GRANTED_ROLE                                                                                | DEF   |
| -----                                                                                       | ----- |
| HRCLERK                                                                                     | YES   |
| HRMANAGER                                                                                   | NO    |
| SQL>                                                                                        |       |

5. Disconnect PDBADMIN from ORCLPDB1.

|                        |  |
|------------------------|--|
| SQL> <b>disconnect</b> |  |
| ...                    |  |
| SQL>                   |  |

## Enable a Non-Default Role

6. Connect to ORCLPDB1 as JGOODMAN. Refer to *Course Practice Environment: Security Credentials* for the **password** value.

|                                                       |  |
|-------------------------------------------------------|--|
| SQL> <b>connect jgoodman/<i>password</i>@orclpdb1</b> |  |
| Connected.                                            |  |
| SQL>                                                  |  |

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

|                                          |  |
|------------------------------------------|--|
| SQL> <b>column role format a30</b>       |  |
| SQL> <b>select * from session_roles;</b> |  |

```
ROLE
```

```

```

```
HRCLERK
```

```
SQL>
```

8. 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>
```

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

```
SQL> select * from session_roles;
```

```
ROLE
```

```

```

```
HRMANAGER
```

```
SQL>
```

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

```
SQL> set role HRMANAGER, HRCLERK;
```

```
Role set.
```

```
SQL>
```

11. 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>
```

12. Exit SQL\*Plus.

```
SQL> exit
...
$
```

13. Exit all terminals.



# **Practices for Lesson 22:**

## **Implementing Oracle Database**

### **Auditing**

## **Practices for Lesson 22: Overview**

---

### **Overview**

In these practices, you will verify Unified Auditing is enabled, create audit users, and create an audit policy.

## Practice 22-1: Enabling Unified Auditing

---

### Overview

In this practice, you enable unified auditing.

### Assumptions

If this practice has been attempted with the current deployment of the practice environment, reset the original state by executing

```
/home/oracle/labs/DBMod_UsersSec/disable_unified_aud.sh.
```

This script removes unified auditing from the Oracle executable, and removes possible unified auditing artifacts created in a prior attempt. Only the `orclcdb` database will be restarted. This script may display errors that can be safely ignored.

### Tasks

1. Open a terminal and set the Oracle environment for the `ORCLCDB` database.

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

2. Shut down all Oracle processes of all instances.

- a. Shut down the listeners, `LISTENER` and `LISTENER2`

```
[oracle@edvmr1p0 ~]$ lsnrctl stop listener
...
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com) (PORT=1521)))
The command completed successfully
[oracle@edvmr1p0 ~]$ lsnrctl stop listener2
...
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com) (PORT=1561)))
The command completed successfully
$
```

- b. Shut down the `orclcdb` instance and exit `sqlplus`.

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

```
SQL> exit
$
```

- c. Verify that all instances are down.

```
$ pgrep -lfa smon
14610 ora_smon_CDBDEV
26743 ora_smon_CDBTEST
$
```

- d. If there are any databases still open change the Oracle environment to that database and shut it down.

```
$. oraenv
ORACLE_SID = [orclcdb] ? CDBDEV
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL> shutdown immediate
...
SQL> exit
...
$. oraenv
ORACLE_SID = [orclcdb] ? CDBTEST
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL> shutdown immediate
...
SQL> exit
...
$
```

- e. Confirm all instances are down with another pgreg –lfa smon command.

```
$ pgrep -lfa smon
$
```

3. Enable the unified auditing feature.

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk uniaud_on ioracle
...
$
```

5. Restart the processes.

- a. Change the Oracle Environment to `orclcdb`.

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

- b. Restart the listener.

```
$ lsnrctl start listener
...
STATUS of the LISTENER

Alias listener
...
$ lsnrctl start listener2
...
STATUS of the LISTENER

Alias listener2
...
$
```

- c. Change your working directory back to `/home/oracle`

```
$ cd
$ pwd
/home/oracle
$
```

- d. Restart the `orclcdb` database instance.

```
$ sqlplus / as sysdba
...
Connected to an idle instance.
SQL> startup
...
SQL>
```

6. Verify that unified auditing is enabled.

```
SQL> column parameter format a20
SQL> column value format a20
SQL> select * from v$option where PARAMETER = 'Unified Auditing';

PARAMETER VALUE CON_ID

```

```
Unified Auditing TRUE 0
SQL>
```

7. Open all PDBs.

```
SQL> alter pluggable database all open;

Pluggable database altered.
SQL>
```

8. Set all databases to open on startup.

```
SQL> alter pluggable database all save state;

Pluggable database altered.
SQL>
```

9. Exit from SQL\*Plus.

```
SQL> exit
...
$
```

10. Exit all terminals.

## Practice 22-2: Creating Audit Users

---

### Overview

In this practice you will create audit users: one account to administer the audit settings and another account to be used by the external auditor. These additional users are optional, but are a good practice that provides a clear separation of duties required in many businesses. In this exercise you will create a common user to administer audit policies and another to be used by the external auditor across all PDBs in the ORCLCDB database.

### Assumptions

Unified auditing has been enabled in the orclcdb database.

### Tasks

1. Open a terminal, use the . oraenv command to source for database orclcdb, then connect to the ORCLCDB instance as a user with SYSDBA privilege.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL>
```

2. Create a database user to be the administrator of the audit settings and policies. Name this user C##AUDMGR. Refer to *Course Practice Environment: Security Credentials* for the **password** value. Assign the AUDIT\_ADMIN role to this user.

```
SQL> create user c##audmgr identified by password container=all;

User created.

SQL> grant connect, audit_admin to c##audmgr container = all;

Grant succeeded.

SQL>
```

3. Create a database user to be used by any person that needs to view the audit data. Name this user C##AUDVWR . Refer to *Practice Environment: Security Credentials* for the **password** value. Assign the AUDIT\_VIEWER role to this user.

```
SQL> create user c##audvwr identified by password container=all;

User created.
```

```
SQL> grant connect, audit_viewer to c##audvwr container=all;
Grant succeeded.
SQL>
```

4. Exit SQL\*Plus and terminals.

```
SQL> exit
...
$ exit
```

5. Close all terminals.

## Practice 22-3: Creating an Audit Policy

---

### Overview

In this practice, as the C##AUDMGR user you will create an audit policy to monitor activity in the HR.JOBS table in the ORCLPDB1 database and apply it to multiple users.

### Assumptions

The C##AUDMGR user has been created. Several users with DML privileges on HR.JOBS have been created.

### Tasks

1. Invoke SQL\*Plus and connect to the ORCLPDB1 database as the C##AUDMGR user. Create a policy named JOBS\_AUDIT\_UPD that audits all auditable statements for the HR.JOBS table.

- a. Open a terminal and set the environment for the orclcdb database by using oraenv.

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

- b. Connect to the ORCLPDB1 PDB as the C##AUDMGR user by using SQL\*Plus. Refer to

*Practice Environment: Security Credentials* for the **password** value.

```
$ sqlplus c##audmgr/password@orclpdb1
...
SQL>
```

- c. Create an audit policy called JOBS\_AUDIT\_UPD to track UPDATE commands issued against the HR.JOBS table.

```
SQL> create audit policy jobs_audit_upd actions update on
 hr.jobs;
```

```
Audit policy created.
```

```
SQL>
```

- d. Enable the audit policy for all users.

```
SQL> audit policy jobs_audit_upd;
```

```
Audit succeeded.
```

```
SQL>
```

2. Verify the creation of the JOBS\_AUDIT\_UPD policy.

```
SQL> column audit_option format A20
SQL> column policy_name format a18
SQL> column object_name format a18
SQL> select policy_name, audit_option, object_name from
audit_unified_policies where policy_name ='JOBS_AUDIT_UPD';

POLICY_NAME AUDIT_OPTION OBJECT_NAME
----- -----
JOBS_AUDIT_UPD UPDATE JOBS
SQL>
```

3. Test the audit policy by connecting as a user that has privileges to update rows in the HR.JOB table.

- a. Connect as the JGOODMAN user and update MAX\_SALARY of the President to \$50000. Be sure to set HRMANAGER role. Refer to *Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect jgoodman/password@orclpdb1
...
SQL> set role HRMANAGER;

Role set.

SQL> desc hr.jobs
Name Null? Type

JOB_ID NOT NULL VARCHAR2(10)
JOB_TITLE NOT NULL VARCHAR2(35)
MIN_SALARY NUMBER(6)
MAX_SALARY NUMBER(6)

SQL> update hr.jobs set max_salary = 50000 where job_title
='President';

1 row updated.

SQL>
```

- b. Connect as the C##AUDMGR user and view the audit trail records for this change. **Note:** Your output may vary from what is shown depending on how many times you have Set Role as the JGOODMAN user. For this practice, you are interested in the row

for the JOBS\_AUDIT\_UPD policy. Refer to *Practice Environment: Security Credentials* for the **password** value.

```
SQL> connect c##audmgr/password@orclpdb1
...
SQL> col dbusername format A8
SQL> col action_name format A8
SQL> col "DATE" format A20
SQL> col unified_audit_policies format a22
SQL> select UNIFIED_AUDIT_POLICIES, DBUSERNAME, ACTION_NAME,
 to_char(EVENT_TIMESTAMP,'DD-MON-YY HH:MI') "DATE"
 from unified_audit_trail
 where DBUSERNAME in ('JGOODMAN')
 and ACTION_NAME not in ('LOGON', 'LOGOFF')
 order by 4;

UNIFIED_AUDIT_POLICIES DBUSERNA ACTION_N DATE

ORA_SECURECONFIG JGOODMAN SET ROLE 21-OCT-20 09:28
ORA_SECURECONFIG JGOODMAN SET ROLE 21-OCT-20 09:30
ORA_SECURECONFIG JGOODMAN SET ROLE 21-OCT-20 09:30
ORA_SECURECONFIG JGOODMAN SET ROLE 21-OCT-20 10:21
JOBS_AUDIT_UPD JGOODMAN UPDATE 21-OCT-20 10:33
```

- c. If you did not see any rows as a result of the query in step 3b, flush the audit records.

**Note:** The default behavior of the Unified Audit Engine is to queue the audit records and write them to the Unified Audit trail as the queue fills. The DBMS\_AUDIT\_MGMT.FLUSH\_UNIFIED\_AUDIT\_TRAIL procedure forces the records in the queue to be written to disk. The audit records are not visible until they are written to the audit trail.

```
SQL> exec sys.dbms_audit_mgmt.flush_unified_audit_trail
```

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

```
SQL>
```

- d. Run the query in step 3b again to view the audit trail records.

4. Exit from SQL\*Plus.

```
SQL> exit
...
$
```

5. Disable unified auditing by running the script,

/home/oracle/labs/DBMod\_UsersSec/disable\_unified\_aud.sh

Note: this script will take several minutes to run.

```
$ $HOME/labs/DBMod_UsersSec/disable_unified_aud.sh
...
$
```

6. Exit all terminals.

# **Practices for Lesson 23: Introduction to Loading and Transporting Data**

## **Practices for Lesson 23**

---

There are no practices for Lesson 23.

# **Practices for Lesson 24: Loading Data**

## **Practices for Lesson 24: Overview**

---

### **Overview**

In these practices, you will use SQL\*Loader to load data.

## Practice 24-1: 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 ORCLPDB1 by using SQL\*Loader in express mode. Data and control files are provided.
- Load data into the SH.INVENTORIES table in ORCLPDB1 by using SQL\*Loader in conventional mode.
- Load data into the SH.INVENTORIES table in ORCLPDB1 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/DBMod_LoadTrans/products.dat` data file into the SH.PRODUCTS table in ORCLPDB1.

1. Open a terminal window and use `oraenv` to set the environment variables for the `orclcdb` database. Use the `dbstart.sh` script to start the database and listener.

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

2. Execute the `$HOME/labs/DBMod_LoadTrans/DP_setup.sh` shell script.  
Note: this script take take apx 2 minutes to run.

```
$ $HOME/labs/DBMod_LoadTrans/DP_setup.sh
...
$
```

3. View the `products.dat` file to learn about its structure.

```
$ cat /home/oracle/labs/DBMod_LoadTrans/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
$
```

4. Start SQL\*Plus and connect to ORCLPDB1 as the SH user. Refer to “Course Practice Environment: Security Credentials” in your Activity Guide for the **password** value.

```
$ sqlplus sh/password@orclpdb1
...
SQL>
```

5. Count the number of rows in the SH.PRODUCTS table. The results indicate that there are seven rows in the table and, therefore, seven products.

```
SQL> select count(*) from sh.products;

COUNT (*)

7

SQL>
```

6. Exit SQL\*Plus.

```
SQL> exit
...
$
```

7. Change to the /home/oracle/labs/DBMod\_LoadTrans directory & verify.

```
$ cd /home/oracle/labs/DBMod_LoadTrans
$ pwd
/home/oracle/labs/DBMod_LoadTrans
$
```

8. Start SQL\*Loader, connect to ORCLPDB1 as the SH user, and load the records from the products.dat file into the SH.PRODUCTS table in ORCLPDB1. The results show that eight rows were successfully loaded. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlldr sh/password@orclpdb1 table=products
SQL*Loader: Release 19.0.0.0.0 - Production on Wed Oct 21 23:24:22
2020
Version 19.3.0.0.0
(c) 1982, 2019, 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.
$
```

9. Start SQL\*Plus and connect to ORCLPDB1 as the SH user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus sh/password@orclpdb1
...
SQL>
```

10. 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 Puerta1 Puerta exterior1
1004 GER Tur1 Auberliche Tur1
1005 FRE Volet1 Volet exterieur1
1006 SPA Obturador1 Obturador exterior1
1007 GER Fenster1 Fensterladen1
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

15 rows selected.
SQL>
```

11. Exit SQL\*Plus.

```
SQL> exit
...
$
```

12. View the products.log file.

```
$ cat products.log

SQL*Loader: Release 19.0.0.0.0 - Production on Wed Oct 21 23:24:22
2020
Version 19.3.0.0.0

(c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Express Mode Load, Table: PRODUCTS
Data File: products.dat
Bad File: products_%p.bad
Discard File: none specified

(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Continuation: none specified
Path used: External Table

Table PRODUCTS, loaded from every logical record.
Insert option in effect for this table: APPEND

 Column Name Position Len Term Encl Datatype
-----+-----+-----+-----+-----+-----+-----+
PRODUCT_ID FIRST * , CHARACTER
COUNTRY NEXT * , CHARACTER
LABEL NEXT * , CHARACTER
DETAILED_LABEL NEXT * , CHARACTER

Generated control file for possible reuse:
OPTIONS (EXTERNAL_TABLE=EXECUTE, TRIM=LRTRIM)
LOAD DATA
INFILE 'products'
APPEND
INTO TABLE PRODUCTS
FIELDS TERMINATED BY ","
(
 PRODUCT_ID,
```

```

COUNTRY,
LABEL,
DETAILED_LABEL
)
End of generated control file for possible reuse.

created temporary directory object SYS_SQLLDR_XT_TMPDIR_00000 for
path /home/oracle/labs/DBMod_LoadTrans

enable parallel DML: ALTER SESSION ENABLE PARALLEL DML

creating external table "SYS_SQLLDR_X_EXT_PRODUCTS"

CREATE TABLE "SYS_SQLLDR_X_EXT_PRODUCTS"
(
 "PRODUCT_ID" NUMBER(6),
 "COUNTRY" CHAR(3),
 "LABEL" VARCHAR2(10),
 "DETAILED_LABEL" VARCHAR2(20)
)
ORGANIZATION external
(
 TYPE oracle_loader
 DEFAULT DIRECTORY SYS_SQLLDR_XT_TMPDIR_00000
 ACCESS PARAMETERS
 (
 RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
 BADFILE 'SYS_SQLLDR_XT_TMPDIR_00000':'products_%p.bad'
 LOGFILE 'products_%p.log_xt'
 READSIZE 1048576
 FIELDS TERMINATED BY "," LRTRIM
 REJECT ROWS WITH ALL NULL FIELDS
 (
 "PRODUCT_ID" CHAR(255),
 "COUNTRY" CHAR(255),
 "LABEL" CHAR(255),
 "DETAILED_LABEL" CHAR(255)
)
)
 location
 (
 'products.dat'

```

```

)
)REJECT LIMIT UNLIMITED

executing INSERT statement to load database table PRODUCTS

INSERT /*+ append parallel(auto) */ INTO PRODUCTS
(
 PRODUCT_ID,
 COUNTRY,
 LABEL,
 DETAILED_LABEL
)
SELECT
 "PRODUCT_ID",
 "COUNTRY",
 "LABEL",
 "DETAILED_LABEL"
FROM "SYS_SQLLDR_X_EXT_PRODUCTS"

dropping external table "SYS_SQLLDR_X_EXT_PRODUCTS"

Table PRODUCTS:
 8 Rows successfully loaded.

Run began on Wed Oct 21 23:24:22 2020
Run ended on Wed Oct 21 23:24:24 2020

Elapsed time was: 00:00:02.10
CPU time was: 00:00:00.02

$
```

13. **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.

14. In the /home/oracle/labs/DBMod\_LoadTrans 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.

```

$ ls -lt products_*.log_xt
-rw-r--r-- 1 oracle oinstall 852 Oct 21 23:24 products_20900.log_xt
$
$ cat products_20900.log_xt
```

```

LOG file opened at 10/21/20 23:24:24

Total Number of Files=1

Data File: products.dat

Log File: products_20900.log_xt

LOG file opened at 10/21/20 23:24:24

Bad File: products_20900.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
 $


```

## Load Data by Using SQL\*Loader in Conventional Mode

In this section, you will load data into the SH.INVENTORIES table in ORCLPDB1 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/DBMod\_LoadTrans.

```

$ cd /home/oracle/labs/DBMod_LoadTrans
$ pwd

```

```
/home/oracle/labs/DBMod_LoadTrans
$
```

2. Start SQL\*Plus and connect to ORCLPDB1 as the SH user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus sh/password@orclpdb1
...
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
...
$
```

5. Start SQL\*Loader, connect to ORCLPDB1 as the SH user, and load the SH.INVENTORIES table from the \$HOME/labs/DBMod\_LoadTrans/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.

```
$ sqlldr userid=sh/password@orclpdb1 control=DP_inventories.ctl
log=inventories.log data=DP_inventories.dat

SQL*Loader: Release 19.0.0.0.0 - Production on Thu Oct 22 00:16:28
2020
Version 19.3.0.0.0

(c) 1982, 2019, 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.
$
```

6. Start SQL\*Plus and connect to ORCLPDB1 as the SH user. Refer to "Course Practice Environment: Security Credentials" for the **password** value.

Hint: use the Linux command line buffer recall using the up arrow.

```
$ sqlplus sh/password@orclpdb1
...
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
...
$
```

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 19.0.0.0.0 - Production on Thu Oct 22 00:16:28
2020
Version 19.3.0.0.0

(c) 1982, 2019, 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 Thu Oct 22 00:16:28 2020

Run ended on Thu Oct 22 00:16:31 2020

Elapsed time was: 00:00:02.97

CPU time was: 00:00:00.02

\$

11. View the end of the control file named `DP_inventories.ctl` in the `cat` command.  
Notice the `APPEND` command.

```
$ cat DP_inventories.ctl
...
-- Load data into the INVENTORIES table
--

LOAD DATA
infile '/home/oracle/labs/DBModLoadTrans/DP_inventories.dat'
INTO TABLE SH.INVENTORIES
APPEND
FIELDS TERMINATED BY ','
(warehouse_id,
product_id,
quantity_on_hand)
$
```

12. Using the `vi` editor, change `APPEND` to `TRUNCATE` so that the control file truncates the table. Save the file and quit the `vi` editor. Hint: to write and quit, use `:wq`

```
-- Oracle Database 19c: Administration Workshop I
-- Oracle Server Technologies - Curriculum Development
--
-- ***Training purposes only***
-- ***Not appropriate for production use***
--

-- Load data into the INVENTORIES table
--

LOAD DATA
infile '/home/oracle/labs/DBModLoadTrans/DP_inventories.dat'
INTO TABLE SH.INVENTORIES
TRUNCATE
FIELDS TERMINATED BY ','
(warehouse_id,
product_id,
quantity_on_hand)
```

13. Start SQL\*Loader, connect to `ORCLPDB1` 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.

```
$ sqlldr userid=sh/password@orclpdb1 control=DP_inventories.ctl
log=inventories.log data=DP_inventories.dat ROWS=10
```

```
SQL*Loader: Release 19.0.0.0.0 - Production on Thu Oct 22 00:28:48
2020
Version 19.3.0.0.0
```

```
(c) 1982, 2019, 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.
$
```

14. Start SQL\*Plus and connect to ORCLPDB1 as the SH user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus sh/password@orclpdb1
...
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/DBMod_LoadTrans/DP_check.sql` SQL script to empty the table and re-enable the check constraint. Then reload the table.

16. Execute the `$HOME/labs/DBMod_LoadTrans/DP_check.sql` script.

```
SQL> @$HOME/labs/DBMod_LoadTrans/DP_check.sql
```

```
Connected.
```

```
Table truncated.
```

```
Table altered.
```

```
Disconnected from Oracle Database 19c Enterprise Edition Release
...
$
```

17. Start SQL\*Loader, connect to `ORCLPDB1` 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.

```
$ sqlldr userid=sh/password@orclpdb1 control=DP_inventories.ctl
log=inventories.log data=DP_inventories.dat ROWS=10
```

```
SQL*Loader: Release 19.0.0.0.0 - Production on Thu Oct 22 00:36:20
2020
Version 19.3.0.0.0
```

```
(c) 1982, 2019, 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.
```

```
$
```

18. 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 19.0.0.0.0 - Production on Thu Oct 22 00:16:28
2020
```

```
Version 19.3.0.0.0
```

```
(c) 1982, 2019, 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.
...
```

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 Thu Oct 22 00:36:20 2020

```
Run ended on Thu Oct 22 00:36:21 2020
```

```
Elapsed time was: 00:00:00.39
CPU time was: 00:00:00.03
$
```

19. **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.

20. Start SQL\*Loader, connect to ORCLPDB1 as the SH user, and load the SH.INVENTORIES table in direct mode. The results indicate that the load completed and the record count is 83. Refer to “Course Practice Environment: Security Credentials” for the *password* value.

```
$ sqlldr userid=sh/password@orclpdb1 control=DP_inventories.ctl
log=inventories.log data=DP_inventories.dat ROWS=10 DIRECT=TRUE
```

```
SQL*Loader: Release 19.0.0.0.0 - Production on Thu Oct 22 00:41:47
2020
```

```
Version 19.3.0.0.0
```

```
(c) 1982, 2019, 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.
$
```

21. **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.

22. **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.

23. **Question:** Does SQL\*Loader in direct mode ignore all constraints?

**Answer:** No, it does not. It enforces `PRIMARY KEY`, `UNIQUE`, and `NOT NULL` constraints.

24. Close the terminal window.



# **Practices for Lesson 25:**

## **Transporting Data**

## **Practices for Lesson 25: Overview**

---

### **Overview**

In these practices, you will move data from one ORCLPDB to another PDB.

## Practice 25-1: Moving Data from One PDB to Another PDB

---

### Overview

In this practice, imagine that you configured ORCLPDB2 with different optimizer parameter values, and you want to test the performance of requests on OE tables in ORCLPDB2 to compare it with the performance of the same queries in ORCLPDB1. Through trial and error, you export all objects from the OE schema from ORCLPDB1 and import them into ORCLPDB2 under a new schema named OETEST for testing purposes.

### Assumptions

You are logged in as the `oracle` user.

### Tasks

#### Export the OE Schema from ORCLPDB1 by Using Data Pump Export

1. Open a new terminal window and use `oraenv` to set the environment variables for the `orclcdb` database.

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

2. Execute the `$HOME/labs/DBMod_LoadTrans/DP_setup.sh` shell script to create tables in ORCLPDB1 and ORCLPDB2.

```
$ $HOME/labs/DBMod_LoadTrans/DP_setup.sh
...
1 row created.

Commit complete.

SQL> SQL> Disconnected from Oracle Database 19c Enterprise Edition
Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
$
```

3. Launch Data Pump export under a connection as OE in ORCLPDB1 to export all objects belonging to OE. Refer to “Course Practice Environment: Security Credentials” in your Activity Guide 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.

```
$ expdp oe/password@orclpdb1 schemas=oe
dumpfile=/u01/app/oracle/admin/orclcdb/dpdump/expoe.dmp

Export: Release 19.0.0.0.0 - Production on Thu Oct 22 00:51:10 2020
Version 19.3.0.0.0

(c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 19c Enterprise Edition Release
19.0.0.0.0 - Production
ORA-39001: invalid argument value
ORA-39000: bad dump file specification
ORA-39088: file name cannot contain a path specification

$
```

4. **Question:** What does the error message lead you to do?

**Answer:** Create a logical directory in ORCLPDB1. 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.

5. Start SQL\*Plus and connect to ORCLPDB1 as the SYS user with the SYSDBA privilege. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus sys/password@orclpdb1 as sysdba
...
SQL>
```

6. Create a Oracle logical directory named DP\_FOR\_OE pointing to '/u01/app/oracle/admin/orclcdb/dpdump'

```
SQL> create directory dp_for_oe as
'/u01/app/oracle/admin/orclcdb/dpdump' ;
```

Directory created.

```
SQL>
```

7. 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>
```

8. Exit SQL\*Plus.

```
SQL> exit
...
$
```

9. Retry the Data Pump export. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ expdp oe/password@orclpdb1 schemas=oe directory=dp_for_oe
dumpfile=expoe.dmp
...
Connected to: Oracle Database 19c Enterprise Edition Release
19.0.0.0.0 - Production
Starting "OE"."SYS_EXPORT_SCHEMA_01": oe/********@orclpdb1
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
Master table "OE"."SYS_EXPORT_SCHEMA_01" successfully
loaded/unloaded

Dump file set for OE.SYS_EXPORT_SCHEMA_01 is:
/u01/app/oracle/admin/orclcdp/dpdump/expoe.dmp
Job "OE"."SYS_EXPORT_SCHEMA_01" successfully completed at Thu Oct
22 00:59:21 2020 elapsed 0 00:01:07

```

10. **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.

11. 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.

```

$ impdp oe/password@orclpdbl schemas=oe directory=dp_for_oe
dumpfile=expoe.dmp sqlfile=oe_SQL
...
Connected to: Oracle Database 19c Enterprise Edition Release
19.0.0.0.0 - Production
Master table "OE"."SYS_SQL_FILE_SCHEMA_01" successfully
loaded/unloaded
Starting "OE"."SYS_SQL_FILE_SCHEMA_01": oe/********@ORCLPDB1
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 Thu Oct
22 01:01:15 2020 elapsed 0 00:00:05

```

12. Review the oe\_SQL.sql script. When you execute the import, all DDL statements are read from the dump file.

```

$ cat /u01/app/oracle/admin/orclcdb/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:F89E917CB8DF9BBF5D97E4E372401916569986C0AA39FB65037583079BE6;T:0
6BBF450C895E197E68CFB0F46690E653EE2C73DF32E8B1C25560F90633935D98FF2
D1F15AB0A0B44BD8F534EFDB3E5851FAE9EF1CA034133C0DDCBCEF1BF867E168165
B881A2928097C64C8F413DC74'
 DEFAULT TABLESPACE "TBS_APP"
 TEMPORARY TABLESPACE "TEMP";
-- new object type path: SCHEMA_EXPORT/SYSTEM_GRANT
GRANT CREATE SESSION TO "OE";
GRANT UNLIMITED TABLESPACE TO "OE";
-- new object type path: SCHEMA_EXPORT/ROLE_GRANT
GRANT "DBA" TO "OE";
-- new object type path: SCHEMA_EXPORT/DEFAULT_ROLE
ALTER USER "OE" DEFAULT ROLE ALL;

```

```

-- new object type path: SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA

BEGIN
 sys.dbms_logrep_imp.instantiate_schema(schema_name=>SYS_CONTEXT('USERENV','CURRENT_SCHEMA'), export_db_name=>'ORCLPDB1',
 inst_scn=>'4052542');
COMMIT;
END;
/
-- new object type path: SCHEMA_EXPORT/SEQUENCE/SEQUENCE
CREATE SEQUENCE "OE"."ORDERS_SEQ" MINVALUE 1 MAXVALUE
999999999999 INCREMENT BY 1 START WITH 10 CACHE 20 NOORDER NOCYCLE
NOKEEP NOSCALE GLOBAL ;
-- new object type path: SCHEMA_EXPORT/TABLE(TABLE)
CREATE TABLE "OE"."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" ;
CREATE TABLE "OE"."ORDERS"
(
 "ORDER_ID" NUMBER(12,0),
 "ORDER_DATE" TIMESTAMP (6) WITH LOCAL TIME ZONE,
 "ORDER_MODE" VARCHAR2(8 BYTE),
 "CUSTOMER_ID" NUMBER(6,0),
 "ORDER_STATUS" NUMBER(2,0),
 "ORDER_TOTAL" NUMBER(12,2),
 "SALES REP_ID" NUMBER(6,0),
 "PROMOTION_ID" NUMBER(6,0)
) 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_APP" ;
-- new object type path: SCHEMA_EXPORT/TABLE/INDEX/INDEX
CREATE INDEX "OE"."I_ORDER_ITEMS" ON "OE"."ORDER_ITEMS"
("ORDER_ID")
 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" PARALLEL 1 ;

ALTER INDEX "OE"."I_ORDER_ITEMS" NOPARALLEL;
-- 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
-- fixup virtual columns...
done fixup virtual columns

$
```

## Import the OE Schema into ORCLPDB2 by Using Data Pump Import

13. Start SQL\*Plus and connect to ORCLPDB2 as the SYSTEM user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus system/password@orclpdb1
...
SQL>
```

14. In case the OETEST schema already exists in ORCLPDB2, 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>
```

15. Exit SQL\*Plus.

```
SQL> exit
...
$
```

16. 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 ORCLPDB2. You will get an error message stating that the directory name for DP\_FOR\_OE is invalid.

```
$ impdp system/password@orclpdb2 remap_schema=oe:oetest
directory=dp_for_oe dumpfile=expoe.dmp

Import: Release 19.0.0.0.0 - Production on Thu Oct 22 01:13:47 2020
Version 19.3.0.0.0

(c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 19c Enterprise Edition Release
19.0.0.0.0 - Production
ORA-39002: invalid operation
ORA-39070: Unable to open the log file.
ORA-39087: directory name DP_FOR_OE is invalid
$
```

17. **Question:** 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 ORCLPDB1, not in ORCLPDB2.

18. Connect to ORCLPDB2 as the SYSTEM user. Refer to “Course Practice Environment: Security Credentials” for the ***password*** value.

```
$ sqlplus system/password@orclpdb2
...
SQL>
```

19. Create the DP\_FOR\_OE directory in ORCLPDB2.

```
SQL> create directory dp_for_oe AS
'/u01/app/oracle/admin/orclcdb/dpdump';

Directory created.

SQL>
```

20. Exit SQL\*Plus.

```
SQL> exit
...
$
```

21. Retry the import operation. Refer to “Course Practice Environment: Security Credentials” for the ***password*** value.

```
$ impdp system/password@orclpdb2 REMAP_SCHEMA=oe:oetest
DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp

Import: Release 19.0.0.0.0 - Production on Thu Oct 22 01:21:44 2020
Version 19.3.0.0.0

(c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 19c Enterprise Edition Release
19.0.0.0.0 - Production
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully
loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01": system/********@orclpdb2
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 Thu
Oct 22 01:22:12 2020 elapsed 0 00:00:27

$
```

22. **Question:** Did the import complete successfully?

**Answer:** Not completely. Data Pump imported only the objects that it could process without any error.

23. **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 ORCLPDB2. The dependent objects of this table, such as an index could not be imported.

24. Create the TBS\_APP2 tablespace in ORCLPDB2.

- Start SQL\*Plus and connect to ORCLPDB2 as the SYSTEM user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```

$ sqlplus system/password@orclpdb2
...
SQL>
```

- b. Issue the CREATE TABLESPACE command to create the TBS\_APP2 tablespace in ORCLPDB2.

```
SQL> create tablespace tbs_app2 datafile
'/u01/app/oracle/oradata/ORCLCDB/orclpdb2/tbs_app02.dbf' size
100m autoextend on next 25m maxsize 500m;

Tablespace created.

SQL>
```

- c. Exit SQL\*Plus.

```
SQL> exit
...
$
```

Retry the import operation. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ impdp system/password@orclpdb2 remap_schema=oe:oetest
directory=dp_for_oe dumpfile=expoe.dmp
...
Connected to: Oracle Database 19c Enterprise Edition Release
19.0.0.0.0 - Production
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully
loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01": system/********@orclpdb2
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 Thu
Oct 22 01:25:54 2020 elapsed 0 00:00:08
$
```

25. **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 ORCLPDB2

Verify that the new OTEST schema exists in ORCLPDB2.

26. Start SQL\*Plus and connect to ORCLPDB2 as the OTEST user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus oetest/password@orclpdb2
...
$
```

27. View the list of tables to which the OTEST user has access.

```
SQL> column table_name format a15
SQL> select table_name from user_tables;

TABLE_NAME

ORDERS
ORDER_ITEMS

SQL>
```

28. 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>
```

29. List the indexes to which the OTEST user has access.

```
SQL> column index_name format a20
SQL> SELECT index_name FROM user_indexes;

INDEX_NAME

SYS_C007956
I_ORDER_ITEMS

SQL>
```

30. List the sequences to which the OTEST user has access.

```
SQL> column sequence_name format a20
SQL> select sequence_name from user_sequences;

SEQUENCE_NAME

ORDERS_SEQ

SQL>
```

31. Exit SQL\*Plus.

```
SQL> exit
...
$
```

32. **Question:** How could you have imported the OE schema from ORCLPDB1 to ORCLPDB2 in one single operation?

**Answer:** The data could be imported from ORCLPDB1 by using a valid database link and written directly back to the connected ORCLPDB2. 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 ORCLPDB2 via a Database Link

33. Start SQL\*Plus and connect to ORCLPDB2 as the SYSTEM user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus system/password@orclpdb2
...
SQL>
```

34. Create a database link in the destination PDB (ORCLPDB2) that will connect to the source PDB (ORCLPDB1). Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
SQL> create database link link_orclpdb1 connect to system
identified by password using 'ORCLPDB1';
```

Database link created.

```
SQL>
```

35. Drop the target user created in the previous import operation.

```
SQL> drop user oetest cascade;
```

User dropped.

```
SQL>
```

36. Exit SQL\*Plus.

```
SQL> exit
...
$
```

37. 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.

```
$ impdp system/password@orclpdb2 SCHEMAS=oe REMAP_SCHEMA=oe:oetest
NETWORK_LINK=link_orclpdb1
...
Starting "SYSTEM"."SYS_IMPORT_SCHEMA_01": system/********@orclpdb2
SCHEMAS=oe REMAP_SCHEMA=oe:oetest NETWORK_LINK=link_orclpdb1
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 Thu
Oct 22 01:47:57 2020 elapsed 0 00:00:53
$
```

38. Verify that the OE schema was imported as OETEST into ORCLPDB2.
- Start SQL\*Plus and connect to ORCLPDB2 as the OETEST user. Refer to “Course Practice Environment: Security Credentials” for the **password** value.

```
$ sqlplus oetest/password@orclpdb2
...
SQL>
```

- View the list of tables to which the OETEST user has access.

```
SQL> column table_name format a20
SQL> select table_name from user_tables;

TABLE_NAME

ORDERS
ORDER_ITEMS

SQL>
```

- Query the number of rows in the ORDER\_ITEMS table. The table has 665 rows.

```
SQL> select count(*) from order_items;

COUNT(*)

665

SQL>
```

- Exit SQL\*Plus.

```
SQL> exit
...
$
```

39. **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 25-2: Transporting a Tablespace

---

### Overview

In this practice, you will transfer a tablespace with all the steps that it would take to transfer it across different platforms (although in your training environment you are using only one host on one platform).

### Assumptions

You have a terminal window open in which you are logged in as the `oracle` OS user. The practice steps indicate when to point to pluggable database `orclpdb1` or `orclpdb2`.

### Tasks

1. Prepare for this practice by executing the `Trans_Tblspc.sh` script from the `$HOME/LABS/DBMod_LoadTrans` directory. This script:

- Creates a new tablespace and user
- As the new user, creates a table and populates it
- Saves its output in the `/tmp/setup.log` file

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ $HOME/labs/DBMod_LoadTrans/Trans_Tblspc.sh
The Oracle base remains unchanged with value /u01/app/oracle
Setup done.
$
```

2. Start a SQL\*Plus session and verify the prerequisites for transporting a tablespace across platforms.
  - a. Log in as the `SYS` user and verify that the source database is in read/write mode.

```
$ sqlplus / as sysdba
...
SQL> column name format a10
SQL> column open_mode format a15
SQL> select name, log_mode, open_mode, current_scn from
v$database;

NAME LOG_MODE OPEN_MODE CURRENT_SCN

ORCLCDB NOARCHIVELOG READ WRITE 4116221
SQL>
```

- b. For performing cross-platform tablespace transport, you must know the exact name of the destination platform to which you are transporting data. Query V\$TRANSPORTABLE\_PLATFORM to view the Linux-based platforms by using the query shown in the code box. In the course practice environment, the Linux x86 64-bit platform is used.

```
SQL> column platform_name format a30
SQL> select platform_id, platform_name, endian_format
 from v$transportable_platform
 where upper(platform_name) like '%LINUX%';

PLATFORM_ID PLATFORM_NAME ENDIAN_FORMAT

10 Linux IA (32-bit) Little
11 Linux IA (64-bit) Little
9 IBM zSeries Based Linux Big
13 Linux x86 64-bit Little
18 IBM Power Based Linux Big
22 Linux OS (S64) Big

6 rows selected.

SQL>
```

- c. Set orclpdb1 as the current container and make the BARTBS tablespace read only. This is required for the export of the tablespace metadata. Then exit SQL\*Plus.

```
SQL> alter session set container=ORCLPDB1;

Session altered.

SQL> alter tablespace bartbs read only;

Tablespace altered.

SQL> exit
...
$
```

3. In the same window, start an RMAN session and connect to your orcl source database as the target instance. Refer to the “Course Practice Environment: Security Credentials” document for the password.

```
$ rman target "'sys@orclpdb1 as sysdba'"
...
target database Password: password
```

```
connected to target database: ORCL:PDB1 (DBID=4064546540)
```

```
RMAN>
```

4. Back up the source tablespace by using the `BACKUP` command with the `TO PLATFORM` clause. Use the `DATAPUMP` clause to indicate that an export dump file for the tablespaces must be created for the tablespace metadata.

```
RMAN> backup to platform 'Linux x86 64-bit' format
 '/u01/app/backup/test.bck' datapump format
 '/u01/app/backup/test.dmp' tablespace bartbs;

Starting backup at 22-OCT-20
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=276 device type=DISK
Running TRANSPORT_SET_CHECK on specified tablespaces
TRANSPORT_SET_CHECK completed successfully

Performing export of metadata for specified tablespaces...
 EXPDP> Starting "SYS"."TRANSPORT_EXP_ORCLCDB_eivf":
 EXPDP> Processing object type
TRANSPORTABLE_EXPORT/STATISTICS/TABLE_STATISTICS
 EXPDP> Processing object type
TRANSPORTABLE_EXPORT/STATISTICS/MARKER
 EXPDP> Processing object type TRANSPORTABLE_EXPORT/PLUGTS_BLK
 EXPDP> Processing object type
TRANSPORTABLE_EXPORT/POST_INSTANCE/PLUGTS_BLK
 EXPDP> Processing object type TRANSPORTABLE_EXPORT/TABLE
 EXPDP> Master table "SYS"."TRANSPORT_EXP_ORCLCDB_eivf"
successfully loaded/unloaded
 EXPDP>

EXPDP> Dump file set for SYS.TRANSPORT_EXP_ORCLCDB_eivf is:
 EXPDP>
/u01/app/oracle/product/19.3.0/dbhome_1/dbs/backup_tts_ORCLCDB_9661
6.dmp
 EXPDP>

EXPDP> Datafiles required for transportable tablespace BARTBS:
```

```

EXPDP> /u01/app/backup/ORCLCDB/orclpdb1/bartbs.dbf
EXPDP> Job "SYS"."TRANSPORT_EXP_ORCLCDB_eivf" successfully
completed at Thu Oct 22 02:07:26 2020 elapsed 0 00:00:35
Export completed

channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00067
name=/u01/app/backup/ORCLCDB/orclpdb1/bartbs.dbf
channel ORA_DISK_1: starting piece 1 at 22-OCT-20
channel ORA_DISK_1: finished piece 1 at 22-OCT-20
piece handle=/u01/app/backup/test.bck tag=TAG20201022T020642
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting full datafile backup set
input Data Pump dump
file=/u01/app/oracle/product/19.3.0/dbhome_1/dbs/backup_tts_ORCLCDB
_96616.dmp
channel ORA_DISK_1: starting piece 1 at 22-OCT-20
channel ORA_DISK_1: finished piece 1 at 22-OCT-20
piece handle=/u01/app/backup/test.dmp tag=TAG20201022T020642
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 22-OCT-20

RMAN>

```

5. Enable read/write operations on the BARTBS tablespace. Then exit RMAN.

```

RMAN> alter tablespace bartbs read write;

Statement processed

RMAN> exit

Recovery Manager complete.
$
```

**Note:** Normally, after you disconnect from the source database, you move the backup sets and the Data Pump export dump file to the destination host by using operating system utilities. *In this training example, you do not need to do it because you only have one host available.*

6. Set your environment variables to point to the `orclcdb` database instance. Then as the `SYS` user, connect to the database by using SQL\*Plus.

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

$ sqlplus / as sysdba
...
SQL> alter session set container=ORCLPDB2;

Session altered.

SQL>
```

7. Create the `BAR` user in `orclpdb2` and grant the `CREATE SESSION` privilege to `BAR`. Exit from SQL\*Plus. Replace `password` with the password specified for this step in the “Course Practice Environment: Security Credentials” document.

```
SQL> create user bar identified by password;

User created.

SQL> grant create session to bar;

Grant succeeded.

SQL> exit
...
$
```

8. In RMAN connect to `orclpdb2`. Use the `RESTORE` command with the `FOREIGN TABLESPACE` clause. The `FORMAT` clause specifies the file destination. Use the `DUMP FILE FROM BACKUPSET` clause to restore the metadata from the dump file, which is required to plug the tablespace into the destination database. Refer to the “Course Practice Environment: Security Credentials” document for the `password`.

```
$ rman target sys@orclpdb2

Recovery Manager: Release 19.0.0.0.0 - Production on Thu Oct 22
02:13:30 2020
Version 19.3.0.0.0

(c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.

target database Password: password
```

```

connected to target database: ORCLCDB:ORCLPDB2 (DBID=1621666632)

RMAN> restore foreign tablespace bartbs FORMAT
' /u01/app/backup/ORCLCDB/orclpdb2/bartbs.dbf' FROM BACKUPSET
' /u01/app/backup/test.bck' DUMP FILE FROM BACKUPSET
' /u01/app/backup/test.dmp';

Starting restore at 22-OCT-20
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=280 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup
set
channel ORA_DISK_1: restoring all files in foreign tablespace
BARTBS
channel ORA_DISK_1: reading from backup piece
/u01/app/backup/test.bck
channel ORA_DISK_1: restoring foreign file 67 to
/u01/app/backup/ORCLCDB/orclpdb2/bartbs.dbf
channel ORA_DISK_1: foreign piece handle=/u01/app/backup/test.bck
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:02
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 Data Pump dump file to
/u01/app/oracle/product/19.3.0/dbhome_1/dbs/backup_tts_ORCLCDB_3806
0.dmp
channel ORA_DISK_1: reading from backup piece
/u01/app/backup/test.dmp
channel ORA_DISK_1: foreign piece handle=/u01/app/backup/test.dmp
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:02

Performing import of metadata...
IMPDP> Master table "SYS"."TSPITR_IMP_ORCLCDB_ht1B" successfully
loaded/unloaded
IMPDP> Starting "SYS"."TSPITR_IMP_ORCLCDB_ht1B":
IMPDP> Processing object type TRANSPORTABLE_EXPORT/PLUGTS_BLK
IMPDP> Processing object type TRANSPORTABLE_EXPORT/TABLE
IMPDP> Processing object type
TRANSPORTABLE_EXPORT/STATISTICS(TABLE_STATISTICS
IMPDP> Processing object type

```

```

TRANSPORTABLE_EXPORT/STATISTICS/MARKER
IMPDP> Processing object type
TRANSPORTABLE_EXPORT/POST_INSTANCE/PLUGTS_BLK
IMPDP> Job "SYS"."TSPITR_IMP_ORCLCDB_ht1B" successfully
completed at Thu Oct 22 02:14:30 2020 elapsed 0 00:00:23
Import completed

Finished restore at 22-OCT-20

RMAN>

```

9. Confirm that the BARTBS tablespace exists in your destination database. Then exit RMAN.

```

RMAN> select tablespace_name, status from dba_tablespaces;

TABLESPACE_NAME STATUS

SYSTEM ONLINE
SYSAUX ONLINE
UNDOTBS1 ONLINE
TEMP ONLINE
USERS ONLINE
TBS_APP ONLINE
TBS_APP2 ONLINE
BARTBS READ ONLY

8 rows selected

RMAN> exit
Recovery Manager complete.
$

```

10. Clean up the practice environment by executing the Trans\_Tblspc\_cleanup.sh script. This script removes the original and the transported tablespace, as well as the backup and dump files. The script saves its output in the /tmp/cleanup.log file.

```

$ $HOME/labs/DBMod_LoadTrans/Trans_Tblspc_cleanup.sh
The Oracle base remains unchanged with value /u01/app/oracle
Cleanup complete.
$

```

11. Exit all terminals.

# **Practices for Lesson 26: Using External Tables to Load and Transport Data**

## **Practices for Lesson 26: Overview**

---

### **Overview**

In these practices, you will query and unload external tables.

## Practice 26-1: Querying External Tables

---

### Overview

In this practice, you query partitioned external tables.

Suppose you received new external files containing records about sales. The sales records are dispatched in two files according to the sales year:

- /home/oracle/labs/DBMod\_LoadTrans/DP\_sales\_1998.dat
- /home/oracle/labs/DBMod\_LoadTrans/DP2\_sales\_1999.dat

You don't want to load or insert the records into a table in ORCLPDB1, rather you want to be able to read the sales data from the external files.

### Assumptions

N/A

### Tasks

1. Open a new terminal window and use `oraenv` to set the environment variables for `orclcdb` database.

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

2. Execute the `$HOME/labs/DBMod_loadTrans/DP_glogin.sh` shell script to set formatting for all columns selected in queries and to place both `.dat` files in DP and DP2 subdirectories.

```
$ $HOME/labs/DBMod_LoadTrans/DP_glogin.sh
The Oracle base remains unchanged with value /u01/app/oracle
$
```

**Note:** You can ignore the error message about not being able to remove the `orders.dmp` file.

3. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. See the “Course Practice Environment: Security Credentials” document in your Activity Guide for the **password**.

```
$ sqlplus system/password@orclpdb1
...
SQL>
```

4. In ORCLPDB1, create the SH.SALES\_EXT\_RANGE external table.
- Create two directories in the database that point to where the external files are stored.

```
SQL> create directory ext_dir as
'/home/oracle/labs/DBMod_LoadTrans/DP/';

Directory created.

SQL> create directory ext_dir2 as
'/home/oracle/labs/DBMod_LoadTrans/DP2/';

Directory created.

SQL>
```

- Create an SH schema for the sales data. See Product-Specific Credentials for the **password**. Grant the SH user CREATE SESSION and CREATE TABLE privileges. Also grant the SH user READ WRITE privileges on the directories you just created (ext\_dir and ext\_dir2).

```
SQL> drop user sh cascade;

User dropped.

SQL> create user sh identified by password;

User created.

SQL> grant create session, create table to sh;

Grant succeeded.

SQL> grant read, write on directory ext_dir to sh;

Grant succeeded.

SQL> grant read, write on directory ext_dir2 to sh;

Grant succeeded.

SQL>
```

- c. In case it already exists, drop the SH.SALES\_EXT\_RANGE table. You should get an error stating that the table does not exist.

```
SQL> drop table sh.sales_ext_range;
drop table sh.sales_ext_range
*
ERROR at line 1:
ORA-00942: table or view does not exist

SQL>
```

- d. View the script \$HOME/labs/DBMod\_LoadTrans/external\_table.sql

```
SQL> ! cat $HOME/labs/DBMod_LoadTrans/external_table.sql
CREATE TABLE sh.sales_ext_range
(time_id DATE NOT NULL,
 prod_id INTEGER NOT NULL,
 cust_id INTEGER NOT NULL,
 channel_id INTEGER NOT NULL,
 promo_id INTEGER NOT NULL,
 quantity_sold NUMBER(10,2),
 amount_sold NUMBER(10,2)
)
ORGANIZATION EXTERNAL
(
 TYPE ORACLE_LOADER
 DEFAULT DIRECTORY ext_dir
 ACCESS PARAMETERS
 (
 RECORDS DELIMITED BY NEWLINE
 BADFILE 'sh%a_%p.bad'
 LOGFILE 'sh%a_%p.log'
 FIELDS TERMINATED BY ','
 MISSING FIELD VALUES ARE NULL
)
)
PARALLEL
REJECT LIMIT UNLIMITED
PARTITION by range (time_id)
(PARTITION year1998 VALUES LESS THAN (TO_DATE('31-12-1998',
'DD-MM-YYYY'))
 LOCATION ('DP_sales_1998.dat'),
 PARTITION year1999 VALUES LESS THAN (TO_DATE('31-12-1999', 'DD-
MM-YYYY'))
 LOCATION (ext_dir2:'DP2_sales_1999.dat'));
```

```
SQL>
```

- e. Execute the following code to create the structure of the external table SH.SALES\_EXT\_RANGE. The code partitions the table on the TIME\_ID column. You can copy the code from \$HOME/labs/DBMod\_LoadTrans/external\_table.sql and paste it into SQL\*Plus.

```
SQL> @$HOME/labs/DBMod_LoadTrans/external_table.sql
```

```
Table created.
```

```
SQL>
```

- f. **Question:** Based on the code in the previous step, which directories does the external table use?

**Answer:** The partitions of the external table use two directories. The default directory for any partition created is ext\_dir. The last partition uses another directory, ext\_dir2, which corresponds to the active files for the current sales.

- g. Verify that the locations are correctly set for the partitions by querying the DBA\_XTERNAL\_LOC\_PARTITIONS view.

```
SQL> select table_name, partition_name, location, directory_name
 from dba_external_loc_partitions;
```

| TABLE_NAME      | PARTITION_NAME | LOCATION           | DIRECTORY_NAME |
|-----------------|----------------|--------------------|----------------|
| SALES_EXT_RANGE | YEAR1998       | DP_sales_1998.dat  |                |
| SALES_EXT_RANGE | YEAR1999       | DP2_sales_1999.dat | EXT_DIR2       |

```
SQL>
```

5. Determine the number of rows in the external table based on specific criteria.

- a. Determine the number of rows for sales in 1998.

```
SQL> select count(*) from sh.sales_ext_range partition
(year1998);
```

```
COUNT (*)
```

```

357668
```

```
SQL>
```

- b. Determine the number of rows for sales in 1999.

```
SQL> select count(*) from sh.sales_ext_range partition
(year1999);

COUNT (*)

495890

SQL>
```

- c. Determine the number of rows for sales in both 1998 and 1999.

```
SQL> SELECT count(*) FROM sh.sales_ext_range;

COUNT (*)

853558

SQL>
```

- d. Exit SQL\*Plus.

```
SQL> exit
...
$
```

6. Issue the following commands to find out whether the number of rows read is equivalent to the number of records that exist in the two external files. The results show that the number of records in the DP\_sales\_1998.dat file is 357675 and the number of records in the DP2\_sales\_1999.dat file is 495899. Together, the number of records equals 853554. This value is higher than the number of rows read, which you found to equal 853558 in the previous step.

```
$ wc -l /home/oracle/labs/DBMod_LoadTrans/DP/DP_sales_1998.dat
357675 /home/oracle/labs/DBMod_LoadTrans/DP/DP_sales_1998.dat
$ wc -l /home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_1999.dat
495899 /home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_1999.dat
```

7. Check the log files to determine the reason for the discrepancy.

a. List the log files. Note: your log file names will be different.

```
$ ls -l /home/oracle/labs/DBMod_LoadTrans/DP/*.log
-rw-r--r-- 1 oracle oinstall 5916 Oct 22 02:51
/home/oracle/labs/DBMod_LoadTrans/DP/sh000_21106.log
-rw-r--r-- 1 oracle oinstall 612 Oct 22 02:51
/home/oracle/labs/DBMod_LoadTrans/DP/sh000_21663.log
-rw-r--r-- 1 oracle oinstall 6222 Oct 22 02:51
/home/oracle/labs/DBMod_LoadTrans/DP/sh001_21108.log
```

b. View the content of all log files. According to the log files, there were 16 records that could not be "inserted" into the external table structure because some fields in the external files contained NULL value whereas the column in the table is set to NOT NULL.

```
$ more /home/oracle/labs/DBMod_LoadTrans/DP/*.log
...
error processing column TIME_ID in row 50000 for datafile
/home/oracle/labs/DBMod_LoadTrans/DP/DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)error processing
column TIME_ID
in row 100000 for datafile
/home/oracle/labs/DBMod_LoadTrans/DP/DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
LOG file opened at 11/24/16 16:32:39
...
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DBMod_LoadTrans/DP/DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DBMod_LoadTrans/DP/
DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DBMod_LoadTrans/DP/
DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DBMod_LoadTrans/DP/
DP_sales_1998.dat
...
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_1999.dat
```

```
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
...
$
```

- Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. See Product-Specific Credentials for the *password*.

```
$ sqlplus system/password@orclpdb1
...
SQL>
```

- Attempt to create an index on the partition key of the external table to get better query performance. The resulting error indicates that you cannot create an index on an external organized table.

```
SQL> create index sh.i_ext_sales_time_id on sh.sales_ext_range
 (time_id);
CREATE INDEX sh.i_ext_sales_time_id ON sh.sales_ext_range (time_id)
*
ERROR at line 1:
ORA-30657: operation not supported on external organized table
SQL>
```

- Suppose that a new file with sales for year 2000 has arrived. Add a new partition called year2000 to the table.

```
SQL> alter table sh.sales_ext_range add partition year2000 values
 less than (TO_DATE('31-12-2000', 'DD-MM-YYYY')) location
 (ext_dir2:'DP2_sales_2000.dat');

Table altered.

SQL>
```

- Count the number of sales rows in the SH.SALES\_EXT\_RANGE table for year 2000. The number of rows read equals 235893.

```
SQL> select count(*) from sh.sales_ext_range partition (year2000);

COUNT (*)

235893

SQL>
```

- Count the actual number of rows in the DP2\_sales\_2000.dat file. Again, the result indicates that the number of rows read (235893) is less than the number of rows in the data file (235898). The discrepancy may or may not be due to null rows getting discarded, as

you observed in preceding steps.

```
SQL> host wc -l
/home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_2000.dat
235898 /home/oracle/labs/DBMod_LoadTrans/DP2/sales_2000.dat

SQL>
```

13. Perform another check on the data. Query the number of rows that have a TIME\_ID value that falls within the year 2000. The results show that the database read only one row.

```
SQL> select count(*) FROM sh.sales_ext_range
where time_id <= to_date('31-12-2000', 'DD-MM-YYYY')
and time_id >= to_date('01-01-2000', 'DD-MM-YYYY');

COUNT (*)

1

SQL>
```

14. Exit SQL\*Plus.

```
SQL> exit
...
$
```

15. View the contents of the DP2\_sales\_2000.dat file. Notice that most records do not contain sales for year 2000. You must ensure that the records satisfy the partitioning conditions. If you were to remedy this situation, you would need to create two distinct files: one for 2000 sales and another one for 2001 sales, and then add another partition for 2001 sales.

```
$ cat /home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_2000.dat
24-OCT-01,135,10792,3,999,1,51.43
24-OCT-01,135,10960,3,999,1,51.43
24-OCT-01,135,11126,3,999,1,51.43
24-OCT-01,135,11136,3,999,1,51.43
24-OCT-01,135,11201,3,999,1,51.43
...
```

16. Close the terminal window.

## Practice 26-2: Unloading External Tables

---

### Overview

In this practice, you will write the OE.ORDERS table data to a dump file using the external tables.

### Assumptions

You completed Practice 4-1 Querying External Tables (only steps 1, 2, and 3 are necessary).

### Tasks

1. Open a terminal window and use oraenv to set the environment variables for the orclcdb database.

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

2. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. See Product-Specific Credentials for the *password*.

```
$ sqlplus system/password@orclpdb1
...
SQL>
```

3. In ORCLPDB1, create an external table called OE.ORDERS\_EXT that unloads the rows from OE.ORDERS to an external file called orders.dmp. Later, that file will be read from an external table in ORCLPDB2.

```
SQL> CREATE TABLE oe.orders_ext
 ORGANIZATION EXTERNAL
 (TYPE ORACLE_DATAPUMP
 DEFAULT DIRECTORY ext_dir
 LOCATION ('orders.dmp'))
 AS SELECT * FROM oe.orders;

Table created.

SQL>
```

4. Verify that the external file (orders.dmp) is listed in the /home/oracle/labs/DBMod\_LoadTrans/DP directory. The result indicates that it is listed. Your date will be different than the one shown below.

```
SQL> HOST ls -l /home/oracle/labs/DBMod_LoadTrans/DP/orders.dmp
-rw-r----- 1 oracle oinstall 16384 Oct 22 03:02
/home/oracle/labs/DBMod_LoadTrans/DP/orders.dmp
SQL>
```

5. Determine the number of rows in the OE.ORDERS\_EXT table

```
SQL> select count(*) from oe.orders_ext;

COUNT (*)

105
SQL>
```

6. Connect to ORCLPDB2 as the SYSTEM user. See Product-Specific Credentials for the *password*.

```
SQL> CONNECT SYSTEM/password@orclpdb2
Connected.
SQL>
```

7. Create a user named OE. If the user exists, drop it first. See “Course Practice Environment: Security Credentials” for the *password*.

```
SQL> drop user oe cascade;

User dropped.

SQL> create user oe identified by password;

User created.

SQL>
```

8. Create an external directory named ext\_dir.

```
SQL> create directory ext_dir as
'/home/oracle/labs/DBMod_LoadTrans/DP/';

Directory created.

SQL>
```

9. Create an external table named OE.ORDERS\_EXT that loads orders.dmp.

```
SQL> create table oe.orders_ext
(order_id NUMBER, order_date TIMESTAMP(6) WITH LOCAL TIME ZONE,
order_mode VARCHAR2(8), customer_id NUMBER(6),
order_status NUMBER(2), order_total NUMBER(8),
sales_rep_id NUMBER(6), promotion_id NUMBER(6))
organization external
(type oracle_loader default directory ext_dir
location ('orders.dmp'));

Table created.
```

```
SQL>
```

10. Try to query the entire OE.ORDERS\_EXT table. You get an error.

```
SQL> select * from oe.orders_ext;
select * from oe.orders_ext
*
ERROR at line 1:
ORA-29913: error in executing ODCIEXTTABLEFETCH callout
ORA-30653: reject limit reached

SQL>
```

11. **Question:** Which type of access driver was used to unload the data from the table into an external file?

**Answer:** The access driver was ORACLE\_DATAPUMP. The binary file (`orders.dmp`) created has the same format as the files used by the Data Pump Import and Export utilities and can be interchanged with them. During the loading (reading from the external table), the same access driver must be used.

12. Re-create the external table with the appropriate access driver.

- a. Drop the OE.ORDERS\_EXT table that you just created.

```
SQL> drop table oe.orders_ext;

Table dropped.

SQL>
```

- b. Create the OE.ORDERS\_EXT table again, and this time, specify TYPE ORACLE\_DATAPUMP (see line 7 below) instead of what you used before, which was TYPE ORACLE\_LOADER.

```
SQL> create table oe.orders_ext
 (order_id number, order_date timestamp(6) with local time zone,
 order_mode VARCHAR2(8), customer_id NUMBER(6),
 order_status NUMBER(2), order_total NUMBER(8),
 sales_rep_id NUMBER(6), promotion_id NUMBER(6))
 organization external
 (type oracle_datapump default directory ext_dir location
 ('orders.dmp'));

Table created.

SQL>
```

13. Query the entire OE.ORDERS\_EXT table again. This time, the query returns 105 rows.

```
SQL> column order_date format a30
SQL> SELECT * FROM oe.orders_ext;

 ORDER_ID ORDER_DATE ORDER_MO CUSTOMER_ID
ORDER_STATUS ORDER_TOTAL SALES_REP_ID PROMOTION_ID
----- -----
0 5458 16-AUG-07 02.34.12.234359 PM direct 101
0 78280 153
1 5397 19-NOV-07 03.41.54.696211 PM direct 102
1 42283 154
1 5454 02-OCT-07 04.49.34.678340 PM direct 103
1 6653 154
...
0 5456 07-NOV-06 08.53.25.989889 PM direct 117
0 3878 163
5 5457 31-OCT-07 10.22.16.162632 PM direct 118
5 21586 159

105 rows selected.

SQL>
```

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

```
SQL> exit
...
$ exit
```

# **Practices for Lesson 27: Automated Maintenance Tasks Overview**

## **Practices for Lesson 27**

---

There are no practices for Lesson 27.

# **Practices for Lesson 28: Managing Tasks and Windows**

## **Practices for Lesson 28: Overview**

---

### **Overview**

In these practices, you manage maintenance tasks and windows.

## Practice 28-1: Enabling and Disabling Automated Maintenance Tasks

---

### Overview

In this practice, you disable and then re-enable an automated maintenance task.

### Tasks

1. Open a terminal window and use oraenv to set the environment variables for the orclcdb database.

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

2. Using sqlplus, log into orclpdb1 as system. See the “Course Practice Environment: Security Credentials” document in your Activity Guide for the **password**. Determine the names of the automated maintenance tasks and the status of each by querying DBA\_AUTOTASK\_CLIENT.

```
$ sqlplus system/password@orclpdb1
...
SQL> column client_name format a40
SQL> select client_name, status from dba_autotask_client;

CLIENT_NAME STATUS

sql tuning advisor ENABLED
auto optimizer stats collection ENABLED
auto space advisor ENABLED

SQL>
```

3. Disable the Automatic SQL Tuning Advisor task by using the DBMS\_AUTO\_TASK\_ADMIN package.

```
SQL> BEGIN
 dbms_auto_task_admin.disable(
 client_name => 'sql tuning advisor',
 operation => NULL,
 window_name => NULL);
END;
/
PL/SQL procedure successfully completed.

SQL>
```

4. Query DBA\_AUTOTASK\_CLIENT again to verify that the task is disabled.

```
SQL> select client_name, status from dba_autotask_client;

CLIENT_NAME STATUS

sql tuning advisor DISABLED
auto optimizer stats collection ENABLED
auto space advisor ENABLED

SQL>
```

5. Re-enable the task by using the DBMS\_AUTO\_TASK\_ADMIN package.

```
SQL> BEGIN
 dbms_auto_task_admin.enable(
 client_name => 'sql tuning advisor',
 operation => NULL,
 window_name => NULL);
END;
/
PL/SQL procedure successfully completed.

SQL>
```

6. Query DBA\_AUTOTASK\_CLIENT to verify that the task is once again enabled.

```
SQL> select client_name, status from dba_autotask_client;

CLIENT_NAME STATUS

sql tuning advisor ENABLED
auto optimizer stats collection ENABLED
auto space advisor ENABLED

SQL>
```

7. Exit SQL\*plus.

```
SQL> exit
...
$
```

## Practice 28-2: Modifying the Duration of a Maintenance Window

---

### Overview

In this practice, you change the duration of the Sunday maintenance window.

### Tasks

1. Open a terminal window and use oraenv to set the environment variables for the orclcdb database.

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

2. Using sqlplus, log into orclpdb1 as sys as sysdba. See the “Course Practice Environment: Security Credentials” document in your Activity Guide for the **password**.

```
$ sqlplus sys/password@orclpdb1 as sysdba
...
SQL>
```

3. Determine the duration of the Sunday maintenance window by querying DBA\_AUTOTASK\_SCHEDULE.

```
SQL> column window_name format a15
SQL> column start_time format a35
SQL> column duration format a15
SQL> select * from dba_autotask_schedule where window_name =
'SUNDAY_WINDOW';

WINDOW_NAME START_TIME DURATION

SUNDAY_WINDOW 25-OCT-20 06.00.00.531459 AM +00:00 +000 20:00:00
SUNDAY_WINDOW 01-NOV-20 06.00.00.531459 AM +00:00 +000 20:00:00
SUNDAY_WINDOW 08-NOV-20 06.00.00.531459 AM +00:00 +000 20:00:00
SUNDAY_WINDOW 15-NOV-20 06.00.00.531459 AM +00:00 +000 20:00:00
SUNDAY_WINDOW 22-NOV-20 06.00.00.531459 AM +00:00 +000 20:00:00

SQL>
```

4. Increase the Sunday maintenance window by 2 hours.
  - a. Use the DBMS\_SCHEDULER.DISABLE subprogram to disable the Sunday window before making changes to it.

```
SQL> BEGIN
 dbms_scheduler.disable(
 name => 'SUNDAY_WINDOW');
END;
/
PL/SQL procedure successfully completed.

SQL>
```

- b. Use the DBMS\_SCHEDULER.SET\_ATTRIBUTE subprogram to increase the Sunday maintenance window by 2 hours.

```
SQL> BEGIN
 dbms_scheduler.set_attribute(
 name => 'SUNDAY_WINDOW',
 attribute => 'DURATION',
 value => numtodsinterval(2, 'hour'));
END;
/
PL/SQL procedure successfully completed.

SQL>
```

- c. Use the DBMS\_SCHEDULER.ENABLE subprogram to re-enable the Sunday window.

```
SQL> BEGIN
 dbms_scheduler.enable(
 name => 'SUNDAY_WINDOW');
END;
/
PL/SQL procedure successfully completed.

SQL>
```

5. Verify that the duration of the Sunday maintenance window has increased by 2 hours by again querying DBA\_AUTOTASK\_SCHEDULE.

```
SQL> select * from dba_autotask_schedule where window_name =
'SUNDAY_WINDOW';
```

| WINDOW_NAME   | START_TIME                   | DURATION             |
|---------------|------------------------------|----------------------|
| SUNDAY_WINDOW | 25-OCT-20 06.00.00.423166 AM | +00:00 +000 02:00:00 |
| SUNDAY_WINDOW | 01-NOV-20 06.00.00.423166 AM | +00:00 +000 02:00:00 |
| SUNDAY_WINDOW | 08-NOV-20 06.00.00.423166 AM | +00:00 +000 02:00:00 |
| SUNDAY_WINDOW | 15-NOV-20 06.00.00.423166 AM | +00:00 +000 02:00:00 |
| SUNDAY_WINDOW | 22-NOV-20 06.00.00.423166 AM | +00:00 +000 02:00:00 |
| SQL>          |                              |                      |

6. Exit sqlplus.

```
SQL> exit
...
$
```

7. Exit all terminals.



# **Practices for Lesson 29: Database Monitoring and Performance Tuning Overview**

## **Practices for Lesson 29**

---

There are no practices for this lesson.

# **Practices for Lesson 30:**

## **Monitoring Database Performance**

## **Practices for Lesson 30: Overview**

---

### **Overview**

In these practices, you will view performance information by using Enterprise Manager Database Express.

# Practice 30-1: 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

### Enable Flash-Based Enterprise Manager Database Express

1. Open a new terminal window and source the `oraenv` script. Use the `dbstart.sh` script to start the database and listener. Run the commands below to enable Flash-based EM Express.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL> @?/rdbms/admin/execemx emx
...
SQL> alter session set container=orclpdb1;
Session altered.

SQL> @?/rdbms/admin/execemx emx
...
SQL> alter session set container=orclpdb2;
Session altered.

SQL> @?/rdbms/admin/execemx emx
...
SQL> exit
...
$
```

## Start an Application Workload

2. Execute the `$HOME/labs/DBMod_MonTune/PERF_setup_tuning.sh` shell script. This script creates a user named `OE`, a tablespace named `TBS_APP`, and a schema named `OE` in the `TBS_APP` tablespace. This script runs for apx several minutes. You can ignore any error messages about objects not existing.

```
$ $HOME/labs/DBMod_MonTune/PERF_setup_tuning.sh
The Oracle base remains unchanged with value /u01/app/oracle
...
412129 rows created.

Commit complete.

$
```

3. Start an application workload in `ORCLPDB1` and `ORCLPDB2`.

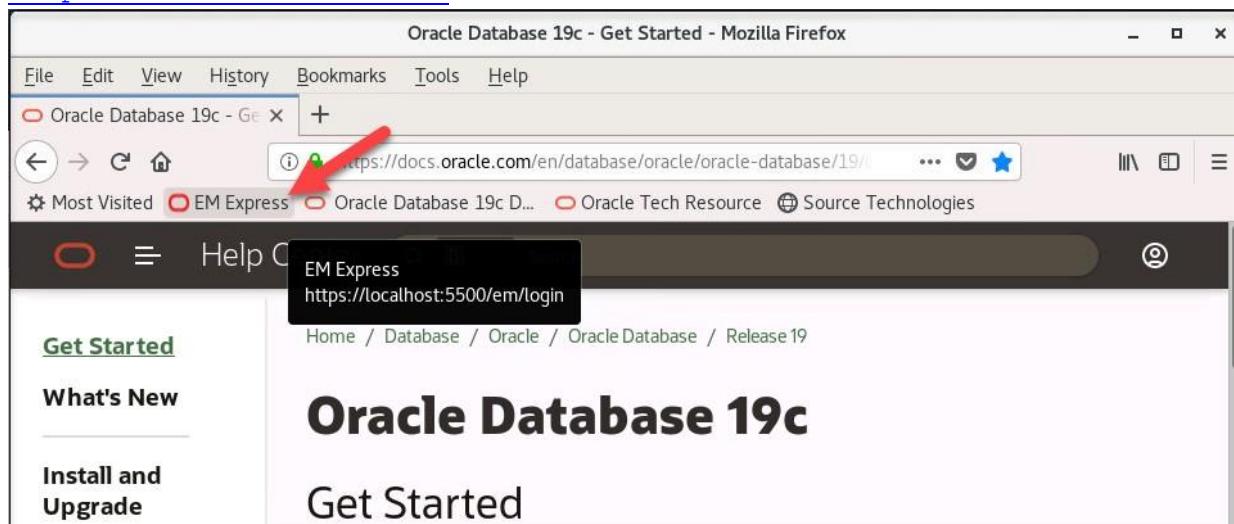
```
$ $HOME/labs/DBMod_MonTune/PERF_loop.sh
...
```

**Note:** This script generates continuous output in the terminal window where it starts.

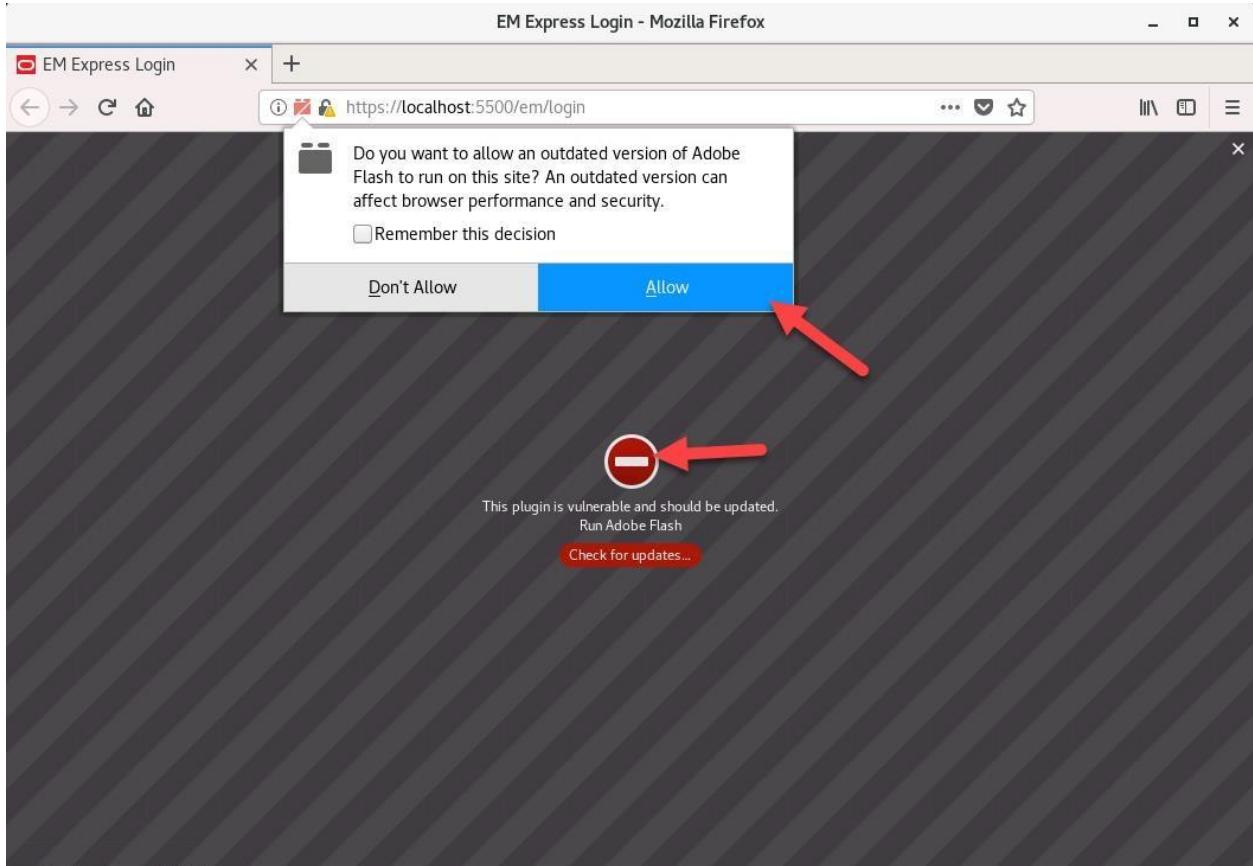
## Review the Performance Hub in EM Express

1. Open a browser. Launch Enterprise Manager Database Express by clicking on the **EM Express** link in the menu-bar –or- by entering the following URL:

<https://localhost:5500/em>



2. If a message appears: This Pluging is vulnerable and should be updated... Click on the red circle and then **Allow** button on the popup. Do not select Remember this decision.



3. On the Login page, enter the username **SYS** and the password. Leave the Container Name box empty, select **as sysdba** and then click **Login**. See the “Course Practice Environment: Security Credentials” document in your Activity Guide for the **password**.

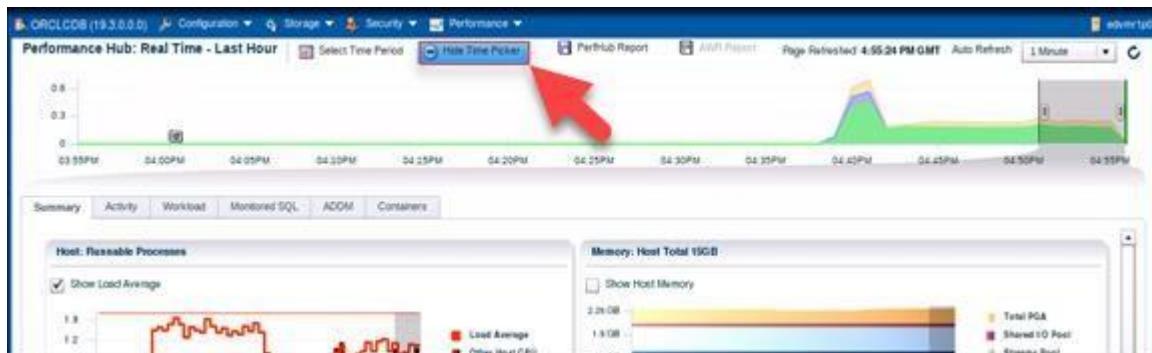
A screenshot of the "Login" page from the EM Express Login application. The page has a "Login" header with a wrench icon. There are three input fields: "User Name" containing "sys", "Password" containing "\*\*\*\*\*", and "Container Name" which is empty. Below these fields is a checked checkbox labeled "as sysdba". At the bottom is a "Login" button.

4. Select Performance and then Performance Hub.

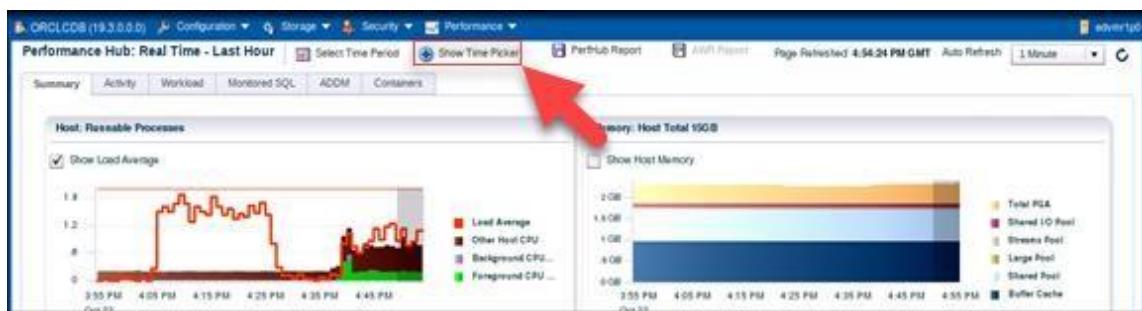


The Performance Hub provides a consolidated view of all performance data for a given time range. You must have the Oracle Diagnostics Pack (licensed option) to use the Performance Hub.

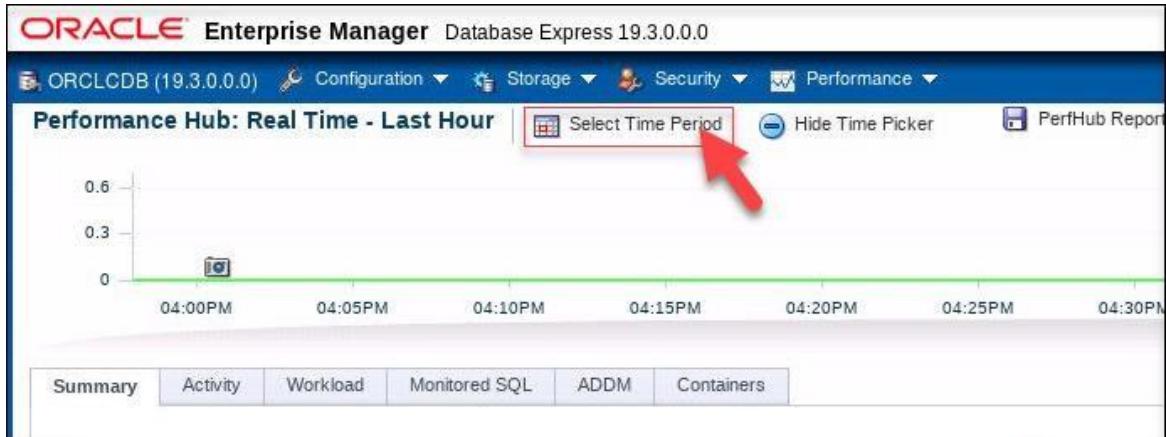
5. Learn about the Time Picker at the top of the page. The Time Picker displays average active sessions over time.  
a. Click **Hide Time Picker** to hide it.



- b. Click **Show Time Picker** to show it.



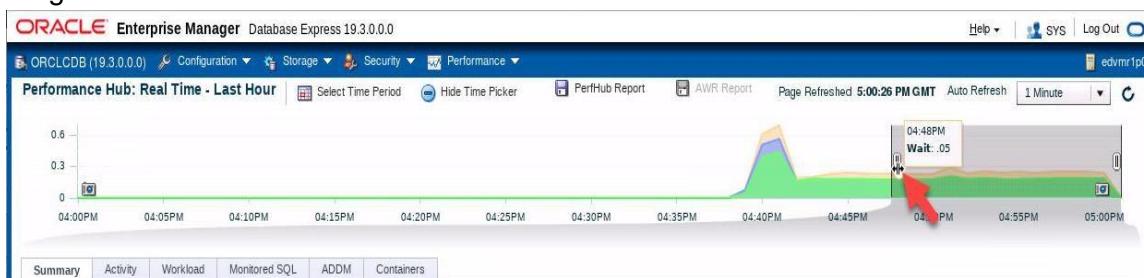
- c. At the top of the page, click **Select Time Period**.



- d. 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.
- e. 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.



- f. 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.



- g. 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 for the database, provided the data is still contained 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. Depending on the uptime of the database, you may receive a message regarding insufficient AWR data when toggling between views.
  - h. Change the time picker back to displaying the last hour.
6. The Performance Hub organizes performance data by dividing it into different tabs. Each tab addresses a specific aspect of database performance.
  - a. 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.
  - b. Click the **Activity** tab. This tab displays Active Session History (ASH) analytics and is available in both real-time and historical mode.
  - c. 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.
  - d. 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.
  - e. 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.
  - f. Click the **Containers** tab. This tab displays performance information about each PDB in the CDB, including active sessions, memory used, I/O requests, and I/O throughput.

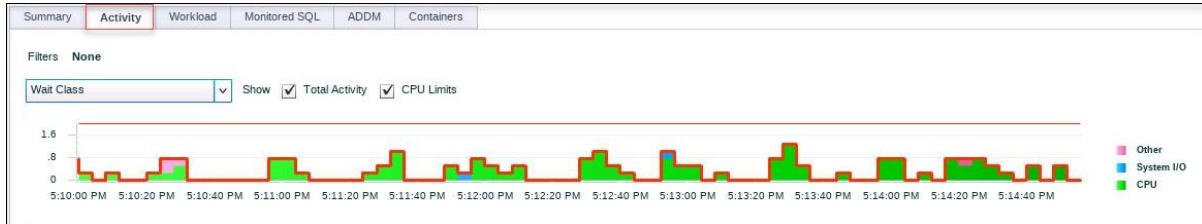
## View Wait Statistics on the Activity Tab

On the Activity tab, you can view the Active Session History (ASH). ASH is part of the Diagnostics and Tuning Pack. It samples information from the [G] V\$ views, allowing you to see current and historical information about active sessions in the database. An active session is a session that is waiting on CPU or any event that does not belong to the IDLE wait class.

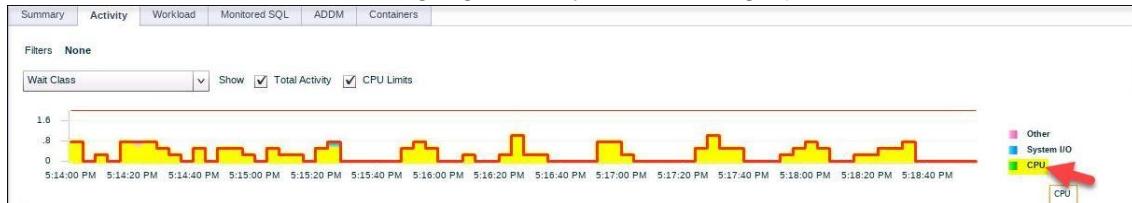
1. If your workload script has ended, start it again in the terminal window.

```
$ $HOME/labs/DBMod_MonTune/PERF_loop.sh
...
```

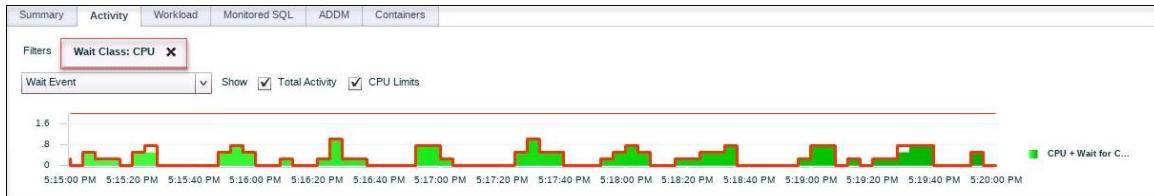
2. In EM Express, click the Refresh button a few times. Eventually you will get some data in the time picker.
3. Click the **Activity** tab.



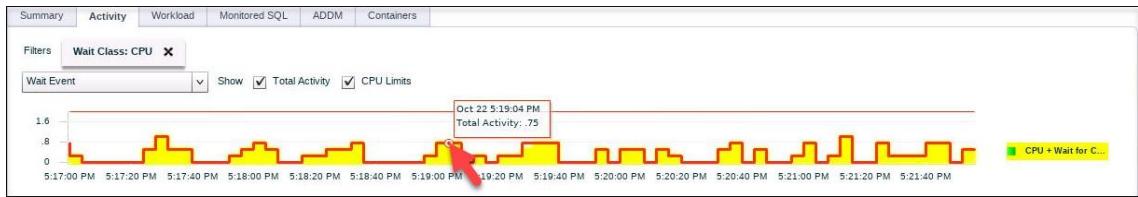
4. View the graph in the middle of the page, which shows wait event information. Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Waits and the associated SQL are key indicators for determining the root cause of an issue.
  - a. 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.
  - b. 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.



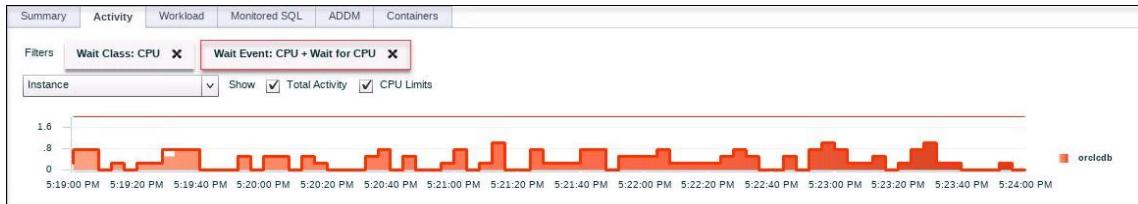
- c. In the legend, click the **CPU** wait class. A filter is created, and the graph drills down into the different wait events for the CPU wait class. Notice that the filters drop-down list is now Wait Event.



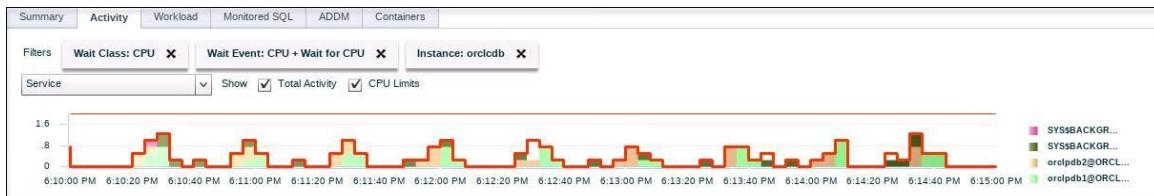
- d. Click the graph. The graph is displaying only one item at the moment, so don't worry about clicking the wrong part. Notice when you positioned your cursor over the graph it turned yellow again.



- e. 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 ORCLCDB instance.



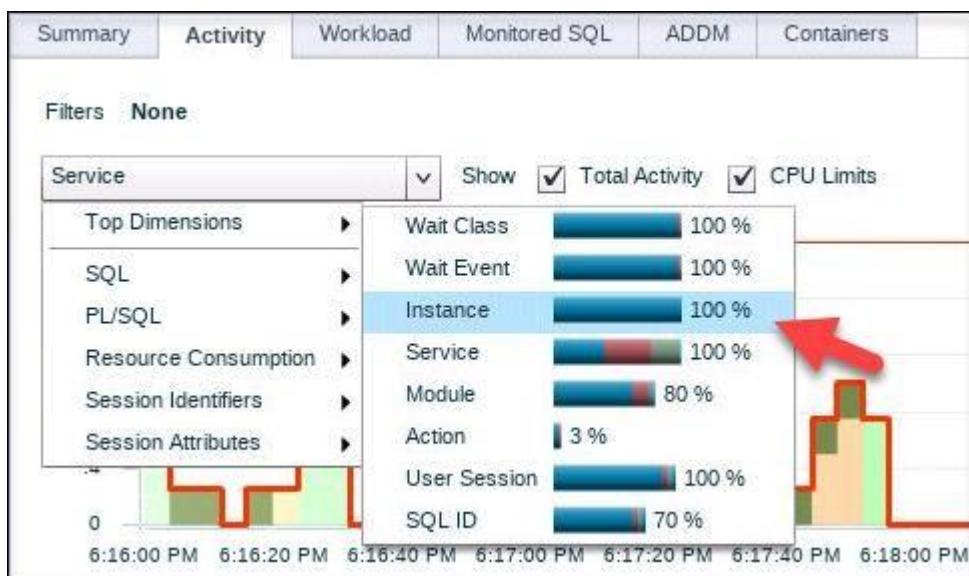
- f. In the legend, click **ORCLCDB**. You have now drilled down into the waits per service; examine the legend on the right side of the graph.



- g. Remove the filters in the graph by clicking the Xs for each filter.

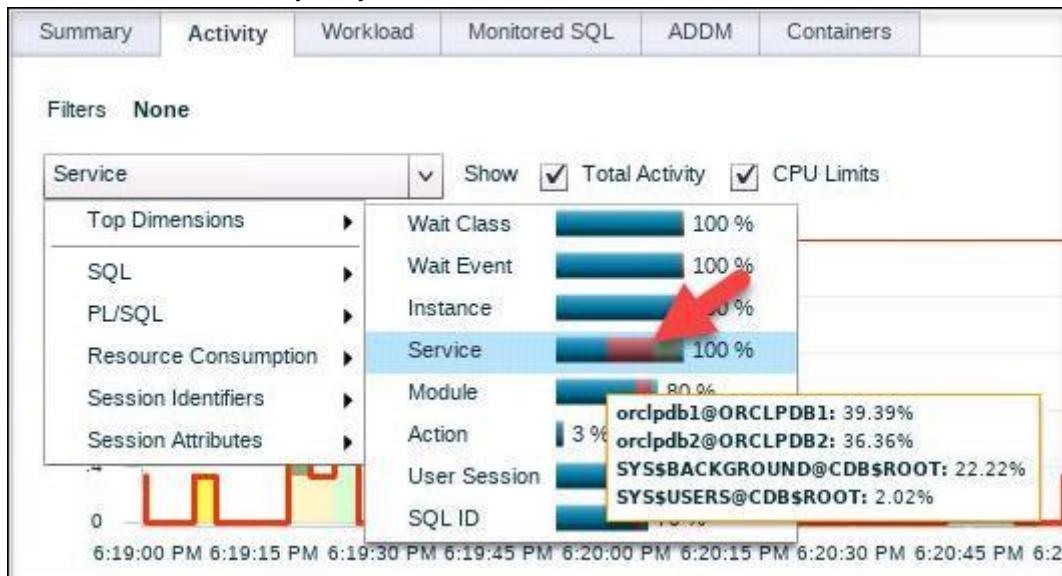


- h. Another way to filter the graph is to select a filter in the drop-down list. In the drop-down list, select **Top Dimensions**.



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.

- i. Position your cursor over **Service** and view the percentage breakdown for service waits. Note values may vary.



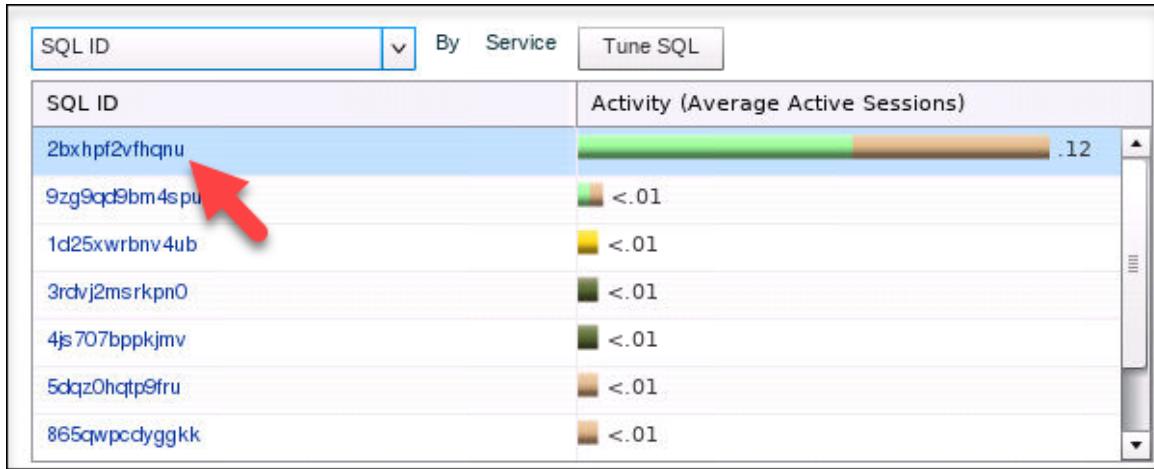
### Filter Wait Statistics for a SQL ID

5. If your workload script has ended, start it again in the terminal window.

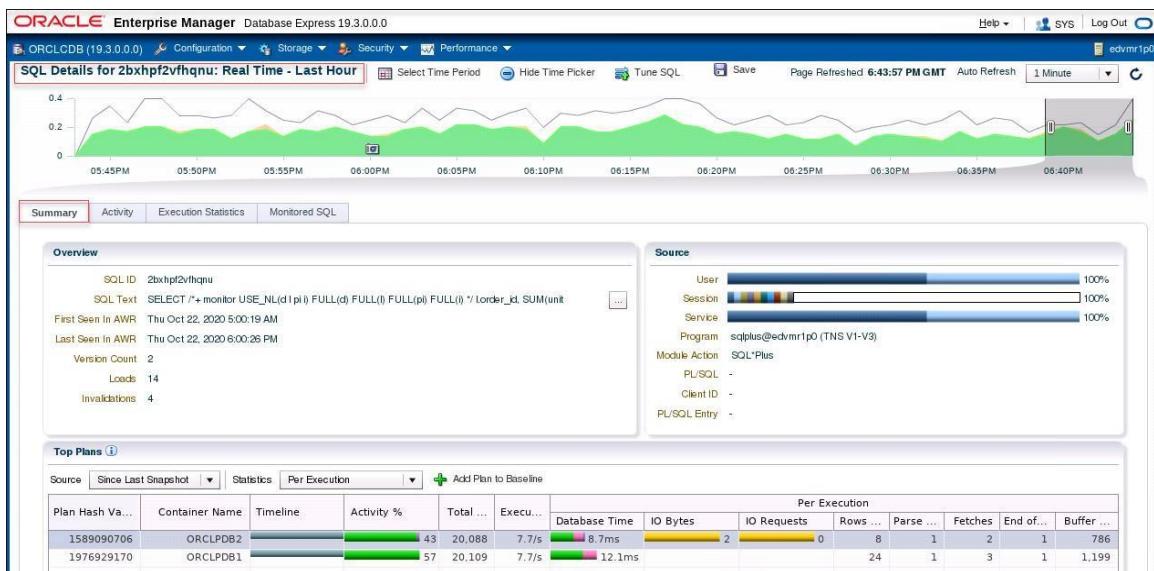
```
$ $HOME/1abs/DBMod_MonTune/PERF_loop.sh
...

```

6. In EM Express, at the bottom left, view the table that shows the activity (average active sessions) for each SQL ID.
- Click the SQL ID that has the greatest average. In this example, the SQL ID is 2bxhpf2vfhqnu.



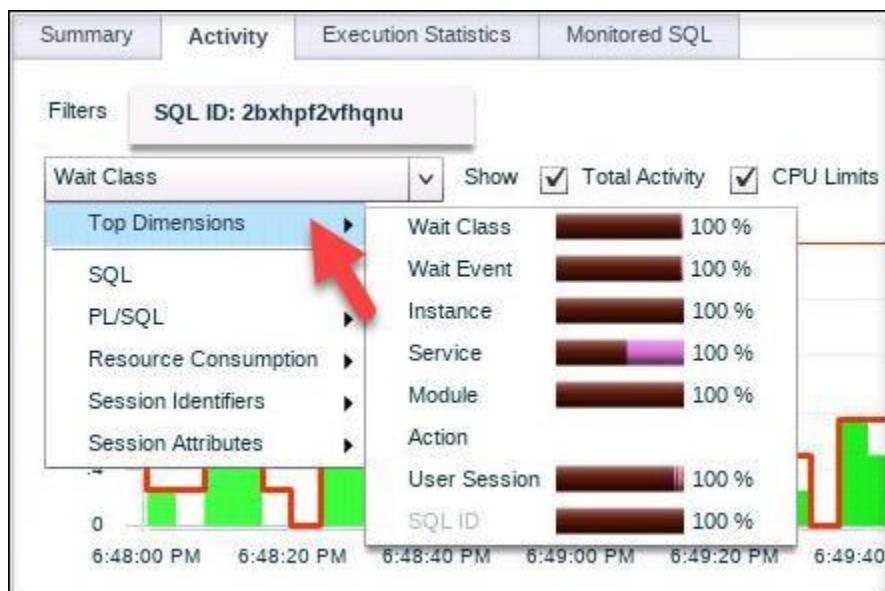
- The Summary tab is displayed with performance information about the selected SQL ID.



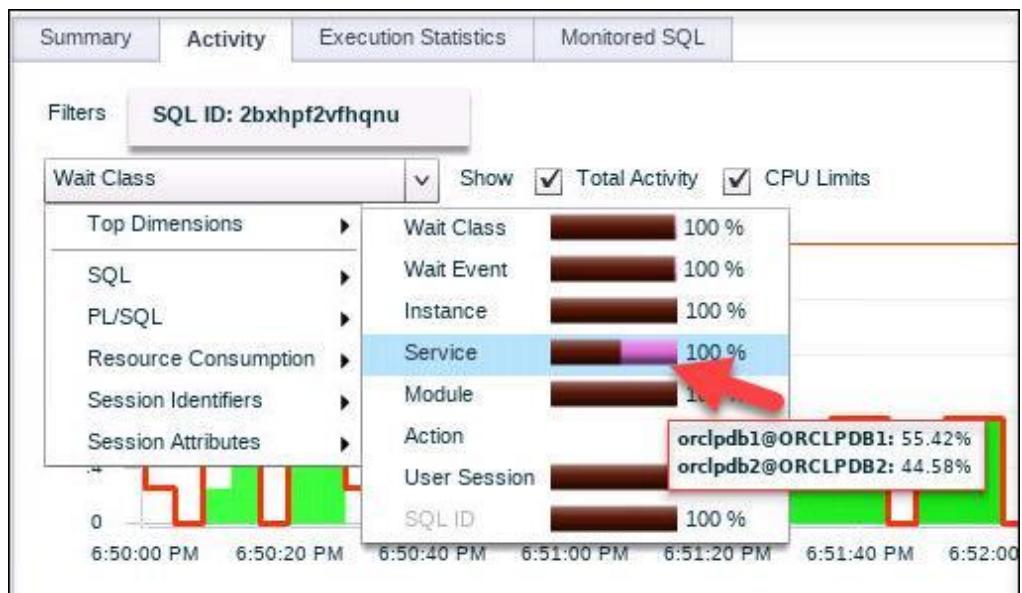
7. Click the **Activity** tab. Notice that the Activity tab is filtered based on your selected SQL ID.



8. In the filters drop-down list, select **Wait Class** and then **Top Dimensions**.



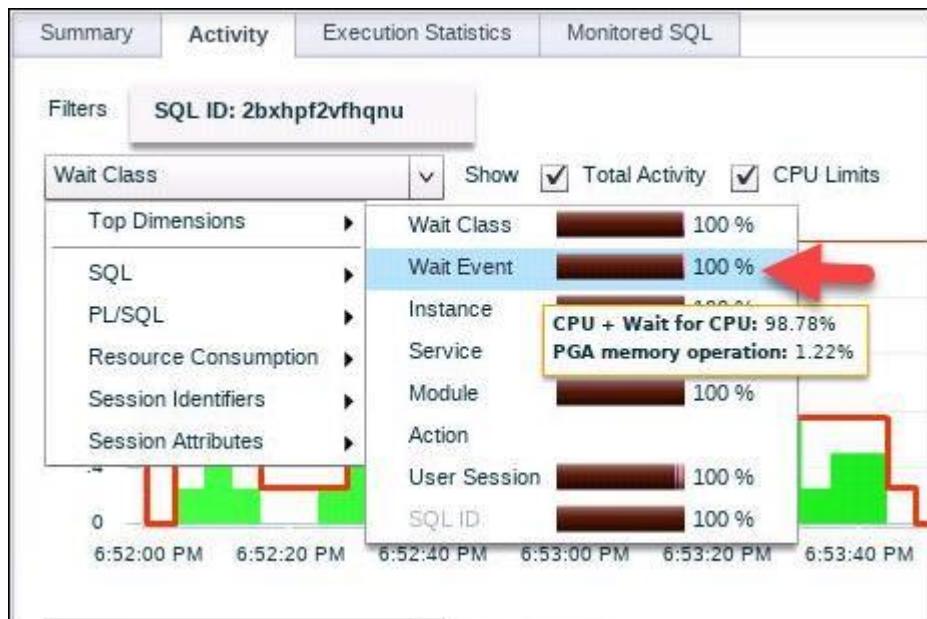
9. Position your cursor over **Service**.



10. Question: What does the information tell you?

Answer: In this example, the percentage value is higher for ORCLPDB1 (55.42%) than ORCLPDB2 (44.58%), which means that executions in ORCLPDB1 have to wait longer than in ORCLPDB2. 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, if the `PERF_loop.sh` script is still running, then press **Ctrl+C** to stop it.  
14. Exit the terminal.

# **Practices for Lesson 31: Processes**

## **Practices for Lesson 31: Overview**

---

In this lesson, you will view the os processes associated with an oracle database.

## Practice 31-1: Examining the Database Background Processes

---

### Overview

In this practice, you use sql and os commands to view the background processes of an oracle database.

### Assumptions

You are logged in as the `oracle` user and the `orclcdb` database is the only running database instance.

### Tasks

1. Open a terminal window and use `oraenv` to set the environment variables for the `orclcdb` database. Use the `dbstart.sh` script to start the database and listener.

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

2. Start SQL\*Plus and connect to `ORCLCDB` database as the `SYS` user with `SYSDBA` privileges .

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

3. Query the `v$instance` view to determine the status of the instance & database, then exit SQL\*Plus.

```
SQL> select instance_name, status, database_status from
v$instance;

INSTANCE_NAME STATUS DATABASE_STATUS

orclcdb OPEN ACTIVE

SQL> exit
...
$
```

4. Use the ps and grep commands to view all the processes associated with the `orclcdb` database. The `ps` command displays information about a selection of the active os processes, while the `-ef` flags associated with the command will do a full listing of all processes. The output of `ps` will be sent to the `grep` command , which is used to filter the output looking for the specific strings, in this case it will be the `orclcdb` database name.

```
$ ps -ef | grep orclcdb
oracle 2856 1 0 Oct29 ? 00:00:01 ora_pmon_orclcdb
oracle 2858 1 0 Oct29 ? 00:00:00 ora_clmn_orclcdb
oracle 2860 1 0 Oct29 ? 00:00:05 ora_psp0_orclcdb
oracle 2895 1 0 Oct29 ? 00:05:19 ora_vktm_orclcdb
oracle 2899 1 0 Oct29 ? 00:00:02 ora_gen0_orclcdb
oracle 2901 1 0 Oct29 ? 00:00:00 ora_mman_orclcdb
oracle 2905 1 0 Oct29 ? 00:00:10 ora_gen1_orclcdb
oracle 2908 1 0 Oct29 ? 00:00:01 ora_diag_orclcdb
oracle 2910 1 0 Oct29 ? 00:00:00 ora_ofsd_orclcdb
oracle 2913 1 0 Oct29 ? 00:00:20 ora_dbrm_orclcdb
oracle 2915 1 0 Oct29 ? 00:00:16 ora_vkrm_orclcdb
oracle 2917 1 0 Oct29 ? 00:00:01 ora_svcb_orclcdb
oracle 2919 1 0 Oct29 ? 00:00:03 ora_pman_orclcdb
oracle 2921 1 0 Oct29 ? 00:00:17 ora_diao_orclcdb
oracle 2923 1 0 Oct29 ? 00:00:05 ora_dbw0_orclcdb
oracle 2925 1 0 Oct29 ? 00:00:04 ora_lgwr_orclcdb
oracle 2927 1 0 Oct29 ? 00:00:08 ora_ckpt_orclcdb
oracle 2929 1 0 Oct29 ? 00:00:03 ora_lg00_orclcdb
oracle 2931 1 0 Oct29 ? 00:00:00 ora_smon_orclcdb
oracle 2933 1 0 Oct29 ? 00:00:00 ora_lg01_orclcdb
oracle 2935 1 0 Oct29 ? 00:00:02 ora_smco_orclcdb
oracle 2937 1 0 Oct29 ? 00:00:00 ora_reco_orclcdb
oracle 2939 1 0 Oct29 ? 00:00:01 ora_w000_orclcdb
oracle 2941 1 0 Oct29 ? 00:00:02 ora_lreg_orclcdb
oracle 2943 1 0 Oct29 ? 00:00:01 ora_w001_orclcdb
oracle 2946 1 0 Oct29 ? 00:00:00 ora_pxmn_orclcdb
oracle 2955 1 0 Oct29 ? 00:00:14 ora_mmon_orclcdb
oracle 2960 1 0 Oct29 ? 00:00:09 ora_mmnl_orclcdb
oracle 2965 1 0 Oct29 ? 00:00:00 ora_d000_orclcdb
oracle 2969 1 0 Oct29 ? 00:00:03 ora_s000_orclcdb
oracle 2973 1 0 Oct29 ? 00:00:05 ora_s001_orclcdb
oracle 2981 1 0 Oct29 ? 00:00:00 ora_tmon_orclcdb
oracle 3064 1 0 Oct29 ? 00:00:33 ora_m000_orclcdb
oracle 3070 1 0 Oct29 ? 00:00:37 ora_m001_orclcdb
oracle 3108 1 0 Oct29 ? 00:00:00 ora_tt00_orclcdb
oracle 3110 1 0 Oct29 ? 00:00:00 ora_tt01_orclcdb
```

```

oracle 3112 1 0 Oct29 ? 00:00:01 ora_tt02_orclcdb
oracle 3141 1 0 Oct29 ? 00:00:00 ora_aqpc_orclcdb
oracle 3148 1 0 Oct29 ? 00:00:00 ora_w002_orclcdb
oracle 3166 1 0 Oct29 ? 00:00:09 ora_p000_orclcdb
oracle 3168 1 0 Oct29 ? 00:00:08 ora_p001_orclcdb
oracle 3170 1 0 Oct29 ? 00:00:02 ora_p002_orclcdb
oracle 3172 1 0 Oct29 ? 00:00:03 ora_p003_orclcdb
oracle 3319 1 0 Oct29 ? 00:00:00 ora_w003_orclcdb
oracle 3378 1 0 Oct29 ? 00:00:54 ora_cjq0_orclcdb
oracle 3454 1 0 Oct29 ? 00:00:00 ora_w004_orclcdb
oracle 3463 1 0 Oct29 ? 00:00:37 ora_m002_orclcdb
oracle 3465 1 0 Oct29 ? 00:00:38 ora_m003_orclcdb
oracle 3468 1 0 Oct29 ? 00:00:41 ora_m004_orclcdb
oracle 3538 1 0 Oct29 ? 00:00:00 ora_qm02_orclcdb
oracle 3561 1 0 Oct29 ? 00:00:00 ora_q005_orclcdb
oracle 11106 1 0 Oct29 ? 00:00:00 ora_w005_orclcdb
oracle 14581 1 0 Oct29 ? 00:00:00 ora_q003_orclcdb
oracle 16634 1 0 00:13 ? 00:00:00 ora_s002_orclcdb
oracle 20893 20891 0 00:32 ? 00:00:00 oracleorclcdb
(DESCRIPTION=(LOCAL=YES) (ADDRESS=(PROTOCOL=TCP) (PORT=1521)))
oracle 22933 1 0 Oct29 ? 00:00:00 ora_w006_orclcdb
oracle 23036 1 0 Oct29 ? 00:00:00 ora_w007_orclcdb
oracle 30569 21226 0 01:09 pts/2 00:00:00 grep --color=auto
orclcdb
$
```

Notice the output included the process id of the grep command.

- You can determine the number of the processes using the pgrep command. pgrep command works like a combination of the ps & grep command. Using the -f flag, it will show only active processes that match the string. Using the -c flag, it will show the count of all processes it found.

```

$ pgrep -lfC orclcdb
56
$
```

- There are certain processes that are critical for a database instance to function, a few of these processes are: pmon, smon, ckpt, lgwr, lreg, mmon Determine if these processes are running.

```

$ pgrep -lf pmon_orclcdb
2856 ora_pmon_orclcd
$
$ pgrep -lf smon_orclcdb
2931 ora_smon_orclcd
```

```
$
$ pgrep -lf lgwr_orclcdb
2925 ora_lgwr_orclcd
$
$ pgrep -lf ckpt_orclcdb
2927 ora_ckpt_orclcd
$
$ pgrep -lf lreg_orclcdb
2941 ora_lreg_orclcd
$
$ pgrep -lf mmon_orclcdb
2955_mmon_orclcd
$
```

7. Some processes can have one or more process performing the function, such as LGWR can have up to 100 sub processes (LG00 -> LG99). The database writer can have up to 100 processes (DBWn where n can be 0 -> 9 and a -> z, then BWnn where nn can be 37 -> 99). Some processes, such as the archiver process (ARC) are optional. Determine if these processes are running using the `ps` command.

- a. Determine how many database writers are running.

[0-9] is a single character wildcard for any character in a list, in this case 0 through 9

```
$ pgrep -lf dbw[0-9]_orclcdb
2923 ora_dbw0_orclcd
$ pgrep -lf bw[0-9][0-9]_orclcdb
$
```

Notice only one Database Writer process , `dbw0`, and no additional process (`DBWn` or `BWnn`).

- b. Determine how many Log Writer sub-processes are running.

[w,0-9] is a single character wildcard for characters: w, 0 through 9 for the first character position, [r,0-9] is a single character wildcard for characters: r, 0 through 9 for the second character position.

```
$ pgrep -lf lg[w,0-9][r,0-9]_orclcdb
2925 ora_lgwr_orclcd
2929 ora_lg00_orclcd
2933 ora_lg01_orclcd
$
```

Notice that three Log Writer processes: `lgwr`, `lg00`, `lg01`

- c. Determine if there are archiver processes arcn (where n is a number 0 through 9).

```
$ pgrep -lf arc[0-9]_orclcdb
$
```

- d. The lack of output shows the archiver process is not running. This can be confirmed by executing the command archiver log list in SQL\*plus

```
$ sqlplus / as sysdba
...
SQL> archive log list
Database log mode No Archive Mode
Automatic archival Disabled
Archive destination USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 29
Current log sequence 31
SQL>
```

8. All the running background processes can be viewed using the v\$bgprocess view.

```
SQL> desc v$bgprocess
Name Null? Type

PADDR RAW(8)
PSERIAL# NUMBER
NAME VARCHAR2(5)
DESCRIPTION VARCHAR2(64)
ERROR NUMBER
TYPE VARCHAR2(5)
PRIORITY VARCHAR2(8)
CON_ID NUMBER
...
SQL> select name,pserial#, description, type from v$bgprocess
order by 1;
NAME PSERIAL# DESCRIPTION TYPE

ABMR 0 Auto BMR Background Process
ACFS 0 ACFS CSS
ACMS 0 Atomic Controlfile to Memory Server
...
CKPT 1 checkpoint
...
DBW0 1 db writer process 0
DBW1 0 db writer process 1
DBW2 0 db writer process 2
DBW3 0 db writer process 3
DBW4 0 db writer process 4
...
LG00 1 Log Writer Slave SLAVE
```

```

LG01 1 Log Writer Slave SLAVE
LGWR 1 Redo etc.
...
LREG 1 Listener Registration
...
MMAN 1 Memory Manager
MMNL 2 Manageability Monitor Process 2
MMON 1 Manageability Monitor Process
...
PMAN 1 process manager
PMON 1 process cleanup
...
RECO 1 distributed recovery
...
SMON 1 System Monitor Process
...
XDMG 0 cell automation manager
XDWK 0 cell automation worker actions

379 rows selected.

SQL>

```

The columns are:

- NAME              Name of this background process in process list
- PSERIAL#        Process state object number, 0 indicates the process is not running, anything else represents that processes is running for the instance.
- DESCRIPTION      Description of the background process
- TYPE              Null for primary processes, otherwise will list SLAVE for sub-processes.

#### 9. Exit SQL\*plus

```

SQL> exit
...
$
```

#### 10. Close the all open terminals.

## Practice 31-2: Identifying the Database Server Processes

---

### Overview

In this practice, you use sql and os commands to view the foreground server processes of an oracle database instance. Clients interact with an oracle database instance using foreground processes, also known as server processes.

### Assumptions

You are logged in as the `oracle` user and only the `orclcdb` database is running.

### Tasks

1. Open a terminal window and use `oraenv` to set the environment variables for the `orclcdb` database. Use the `dbstart.sh` script to start the database and listener.

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

2. Start SQL\*Plus and connect to ORCLCDB database as the `SYS` user with `SYSDBA` privileges. Refer to *Practice Environment: Security Credentials* for the `password` value.

```
$ sqlplus sys/password@orclcdb as sysdba
...
SQL>
```

3. Examine the `v$session` view

```
SQL> desc v$session
```

| Name     | Null? | Type          |
|----------|-------|---------------|
| SADDR    |       | RAW(8)        |
| SID      |       | NUMBER        |
| SERIAL#  |       | NUMBER        |
| AUDSID   |       | NUMBER        |
| PADDR    |       | RAW(8)        |
| USER#    |       | NUMBER        |
| USERNAME |       | VARCHAR2(128) |
| COMMAND  |       | NUMBER        |
| OWNERID  |       | NUMBER        |
| TADDR    |       | VARCHAR2(16)  |
| LOCKWAIT |       | VARCHAR2(16)  |
| STATUS   |       | VARCHAR2(8)   |
| SERVER   |       | VARCHAR2(9)   |
| SCHEMA#  |       | NUMBER        |

|                           |                |
|---------------------------|----------------|
| SCHEMANAME                | VARCHAR2 (128) |
| OSUSER                    | VARCHAR2 (128) |
| PROCESS                   | VARCHAR2 (24)  |
| MACHINE                   | VARCHAR2 (64)  |
| PORT                      | NUMBER         |
| TERMINAL                  | VARCHAR2 (30)  |
| PROGRAM                   | VARCHAR2 (48)  |
| TYPE                      | VARCHAR2 (10)  |
| SQL_ADDRESS               | RAW (8)        |
| SQL_HASH_VALUE            | NUMBER         |
| SQL_ID                    | VARCHAR2 (13)  |
| SQL_CHILD_NUMBER          | NUMBER         |
| SQL_EXEC_START            | DATE           |
| SQL_EXEC_ID               | NUMBER         |
| PREV_SQL_ADDR             | RAW (8)        |
| PREV_HASH_VALUE           | NUMBER         |
| PREV_SQL_ID               | VARCHAR2 (13)  |
| PREV_CHILD_NUMBER         | NUMBER         |
| PREV_EXEC_START           | DATE           |
| PREV_EXEC_ID              | NUMBER         |
| PLSQL_ENTRY_OBJECT_ID     | NUMBER         |
| PLSQL_ENTRY_SUBPROGRAM_ID | NUMBER         |
| PLSQL_OBJECT_ID           | NUMBER         |
| PLSQL_SUBPROGRAM_ID       | NUMBER         |
| MODULE                    | VARCHAR2 (64)  |
| MODULE_HASH               | NUMBER         |
| ACTION                    | VARCHAR2 (64)  |
| ACTION_HASH               | NUMBER         |
| CLIENT_INFO               | VARCHAR2 (64)  |
| FIXED_TABLE_SEQUENCE      | NUMBER         |
| ROW_WAIT_OBJ#             | NUMBER         |
| ROW_WAIT_FILE#            | NUMBER         |
| ROW_WAIT_BLOCK#           | NUMBER         |
| ROW_WAIT_ROW#             | NUMBER         |
| TOP_LEVEL_CALL#           | NUMBER         |
| LOGON_TIME                | DATE           |
| LAST_CALL_ET              | NUMBER         |
| PDML_ENABLED              | VARCHAR2 (3)   |
| FAILOVER_TYPE             | VARCHAR2 (13)  |
| FAILOVER_METHOD           | VARCHAR2 (10)  |
| FAILED_OVER               | VARCHAR2 (3)   |
| RESOURCE_CONSUMER_GROUP   | VARCHAR2 (32)  |

|                               |               |
|-------------------------------|---------------|
| PDML_STATUS                   | VARCHAR2 (8)  |
| PDDL_STATUS                   | VARCHAR2 (8)  |
| PQ_STATUS                     | VARCHAR2 (8)  |
| CURRENT_QUEUE_DURATION        | NUMBER        |
| CLIENT_IDENTIFIER             | VARCHAR2 (64) |
| BLOCKING_SESSION_STATUS       | VARCHAR2 (11) |
| BLOCKING_INSTANCE             | NUMBER        |
| BLOCKING_SESSION              | NUMBER        |
| FINAL_BLOCKING_SESSION_STATUS | VARCHAR2 (11) |
| FINAL_BLOCKING_INSTANCE       | NUMBER        |
| FINAL_BLOCKING_SESSION        | NUMBER        |
| SEQ#                          | NUMBER        |
| EVENT#                        | NUMBER        |
| EVENT                         | VARCHAR2 (64) |
| P1TEXT                        | VARCHAR2 (64) |
| P1                            | NUMBER        |
| P1RAW                         | RAW (8)       |
| P2TEXT                        | VARCHAR2 (64) |
| P2                            | NUMBER        |
| P2RAW                         | RAW (8)       |
| P3TEXT                        | VARCHAR2 (64) |
| P3                            | NUMBER        |
| P3RAW                         | RAW (8)       |
| WAIT_CLASS_ID                 | NUMBER        |
| WAIT_CLASS#                   | NUMBER        |
| WAIT_CLASS                    | VARCHAR2 (64) |
| WAIT_TIME                     | NUMBER        |
| SECONDS_IN_WAIT               | NUMBER        |
| STATE                         | VARCHAR2 (19) |
| WAIT_TIME_MICRO               | NUMBER        |
| TIME_REMAINING_MICRO          | NUMBER        |
| TIME_SINCE_LAST_WAIT_MICRO    | NUMBER        |
| SERVICE_NAME                  | VARCHAR2 (64) |
| SQL_TRACE                     | VARCHAR2 (8)  |
| SQL_TRACE_WAITS               | VARCHAR2 (5)  |
| SQL_TRACE_BINDS               | VARCHAR2 (5)  |
| SQL_TRACE_PLAN_STATS          | VARCHAR2 (10) |
| SESSION_EDITION_ID            | NUMBER        |
| CREATOR_ADDR                  | RAW (8)       |
| CREATOR_SERIAL#               | NUMBER        |
| ECID                          | VARCHAR2 (64) |
| SQL_TRANSLATION_PROFILE_ID    | NUMBER        |

|                          |                 |
|--------------------------|-----------------|
| PGA_TUNABLE_MEM          | NUMBER          |
| SHARD_DDL_STATUS         | VARCHAR2 (8)    |
| CON_ID                   | NUMBER          |
| EXTERNAL_NAME            | VARCHAR2 (1024) |
| PLSQL_DEBUGGER_CONNECTED | VARCHAR2 (5)    |
| SQL>                     |                 |

4. Determine the SESSION\_ID of your current session by using a sys\_content environment query. This will be used to query the V\$SESSION view. Note: your value of SESSION\_ID may be different.

```
SQL> column session_id format a20
SQL> select sys_context('userenv', 'sessionid') Session_ID from
dual;

SESSION_ID

4294967295

SQL>
```

5. Use the SESSION\_ID to view details of your session:

```
SQL> column username format a11
SQL> column osuser format a11
SQL> column machine format a11
SQL> column process format a11
SQL> column program format a30
SQL> select username, osuser, machine, process, program, port
from v$session where AUDSID=4294967295 ;

USERNAME OSUSER MACHINE PROCESS PROGRAM
PORT

SYS oracle edvmr1p0 24117 sqlplus@edvmr1p0
(TNS V1-V3) 51468
```

The PROCESS column shows the OS Process id, referred to as PID, and the PORT column displays the port number used for the connection. **Note:** In this example, Oracle Networking connected through the listener and was assigned port 51468 for the session. The value for both your PORT and PROCESS may be different.

6. Use the `ps` command to display the information about the OS Process Id for your session.

```
SQL> host
$ ps -ef | grep 24117
oracle 24117 11237 0 03:31 pts/3 00:00:00 sqlplus as
sysdba
oracle 28395 24117 0 04:05 pts/3 00:00:00 /bin/bash
oracle 28844 28395 0 04:05 pts/3 00:00:00 grep --color=auto
24117
$ exit
SQL>
```

Process Id 24117 is a the sqlplus / as sysdba

Process ID 28395 is a child process of 24117, it is the os bash session spawned with the host command.

7. Exit SQL\*plus.

```
SQL> exit
...
$
```

8. Close all terminals.



# **Practices for Lesson 32: Tuning Database Memory**

## **Practices for Lesson 32: Overview**

---

### **Overview**

In this practice, you will view the memory configuration of a database.

## Practice 32-1: Viewing Memory Configurations

---

### Overview

In this practice, you use SQL\*Plus to query various views concerning instance memory configuration.

### Assumptions

You are logged in as the `oracle` user and the `orclcdb` database is running.

### Tasks

1. Open a new terminal window and source the `oraenv` script, then log into `sqlplus` as `sysdba`.

```
$. oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL>
```

2. Determine if the database is using Automatic Memory Management (AMM) or Automatic Shared Memory Management (ASMM).

```
SQL> column name format a30
SQL> column value format a20
SQL> select name, value from v$parameter where name in
('sga_target','sga_max_size','memory_target',
'memory_max_target') order by 1;

NAME VALUE

memory_max_target 0
memory_target 0
sga_max_size 2013265920
sga_target 2013265920

SQL>
```

With no values set for `memory_target` and `memory_max_target`, the database is not using Automatic Memory Management (AMM). The values in `sga_target` and `sga_max_size` indicate that Automatic Shared Memory Management (ASMM) is in use.

3. Determine if there are any setting for the various memory pools/caches in the sga.

```
SQL> select name, value from v$parameter where name in
 ('shared_pool_size','large_pool_size','db_cache_size','java_pool_
size','streams_pool_size');

NAME VALUE

shared_pool_size 0
large_pool_size 0
java_pool_size 0
streams_pool_size 0
db_cache_size 0

SQL>
```

The values of zero indicates that the database instance is managing the memory in the caches/pools automatically. If any of those contained a nonzero value, the value is used as the minimum setting for that component where the database instance cannot dynamically decrease the size of the pool/cache below that value. However, the database instance can dynamically increase the component to sizes above that specified value.

4. Examine the view v\$memory\_dynamic\_components

```
SQL> desc v$memory_dynamic_components

Name Null? Type

COMPONENT VARCHAR2(64)
CURRENT_SIZE NUMBER
MIN_SIZE NUMBER
MAX_SIZE NUMBER
USER_SPECIFIED_SIZE NUMBER
OPER_COUNT NUMBER
LAST_OPER_TYPE VARCHAR2(13)
LAST_OPER_MODE VARCHAR2(9)
LAST_OPER_TIME DATE
GRANULE_SIZE NUMBER
CON_ID NUMBER

SQL>
```

5. Determine how much memory is assigned to each pool/cache in the instance by querying the view v\$memory\_dynamic\_components

| COMPONENT<br>LAST_OPER_TYP            | CURRENT_SIZE | MIN_SIZE   | MAX_SIZE   |
|---------------------------------------|--------------|------------|------------|
| unified pga pool<br>STATIC            | 0            | 0          | 0          |
| streams pool<br>GROW                  | 33554432     | 16777216   | 33554432   |
| shared pool<br>GROW                   | 687865856    | 637534208  | 687865856  |
| memoptimize buffer cache<br>STATIC    | 0            | 0          | 0          |
| large pool<br>SHRINK                  | 16777216     | 16777216   | 33554432   |
| java pool<br>STATIC                   | 0            | 0          | 0          |
| Shared IO Pool<br>STATIC              | 100663296    | 100663296  | 100663296  |
| SGA Target<br>STATIC                  | 2013265920   | 2013265920 | 2013265920 |
| RECYCLE buffer cache<br>STATIC        | 0            | 0          | 0          |
| PGA Target<br>STATIC                  | 671088640    | 671088640  | 671088640  |
| KEEP buffer cache<br>STATIC           | 0            | 0          | 0          |
| In-Memory Area<br>STATIC              | 0            | 0          | 0          |
| In Memory RW Extension Area<br>STATIC | 0            | 0          | 0          |
| In Memory RO Extension Area<br>STATIC | 0            | 0          | 0          |
| Data Transfer Cache<br>STATIC         | 0            | 0          | 0          |
| DEFAULT buffer cache<br>GROW          | 1157627904   | 1140850688 | 1224736768 |
| DEFAULT 8K buffer cache<br>STATIC     | 0            | 0          | 0          |

```

DEFAULT 4K buffer cache 0 0 0
STATIC

DEFAULT 32K buffer cache 0 0 0
STATIC

DEFAULT 2K buffer cache 0 0 0
STATIC

DEFAULT 16K buffer cache 0 0 0
STATIC

ASM Buffer Cache 0 0 0
STATIC

22 rows selected.

SQL>

```

6. Both ASMM and AMM update the spfile with the current memory configuration information. Create a readable version of the spfile to view the contents.

```

SQL> create pfile='/tmp/initorclcdb.ora' from spfile;

File created.

SQL>

```

7. Display the contents of /tmp/initorclcdb.ora using the cat command.

```

SQL> ! cat /tmp/initorclcdb.ora
orclcdb._data_transfer_cache_size=0
orclcdb._db_cache_size=1157627904
orclcdb._inmemory_ext_roarea=0
orclcdb._inmemory_ext_rwarea=0
orclcdb._java_pool_size=0
orclcdb._large_pool_size=16777216
orclcdb._oracle_base='/u01/app/oracle'#ORACLE_BASE set from
environment
orclcdb._pga_aggregate_target=671088640
orclcdb._sga_target=2013265920
orclcdb._shared_io_pool_size=100663296
orclcdb._shared_pool_size=687865856
orclcdb._streams_pool_size=33554432
orclcdb._unified_pga_pool_size=0
*.audit_file_dest='/u01/app/oracle/admin/orclcdb/adump'
*.audit_trail='db'
*.compatible='19.0.0'
*.control_files='/u01/app/oracle/oradata/ORCLCDB/control01.ctl','
/u01/app/oracle/fast recovery area/ORCLCDB/control02.ctl'

```

```

*.db_block_size=8192
*.db_name='orclcdb'
*.db_recovery_file_dest='/u01/app/oracle/fast_recovery_area'
*.db_recovery_file_dest_size=14970m
*.diagnostic_dest='/u01/app/oracle'
*.disk_asynch_io=FALSE
*.dispatchers='(PROTOCOL=tcp) (MULTIPLEX=on)'
*.enable_pluggable_database=true
*.job_queue_processes=15
*.local_listener='LISTENER_ORCLCDB,LISTENER2'
*.nls_language='AMERICAN'
*.nls_territory='AMERICA'
*.open_cursors=300
*.optimizer_use_sql_plan_baselines=TRUE
*.pga_aggregate_target=640m
*.processes=300
*.remote_listener='LISTENER_CMAN'
*.remote_login_passwordfile='EXCLUSIVE'
*.sec_max_failed_login_attempts=2#Reduce for tighter security.
*.sga_target=1920m
*.shared_servers=3
*.undo_tablespace='UNDOTBS1'

SQL>

```

The parameters that begin with a double underscore (highlighted in red above) are the current memory pool/cache sizes. Oracle updates the spfile file every time it changes memory so that if the database is shutdown and restarted, the instance would not have to relearn the optimal memory configuration and will continue with the last memory configuration.

#### 8. Describe the view v\$memory\_resize\_ops

| SQL> desc v\$memory_resize_ops |       |               |
|--------------------------------|-------|---------------|
| Name                           | Null? | Type          |
| COMPONENT                      |       | VARCHAR2 (64) |
| OPER_TYPE                      |       | VARCHAR2 (13) |
| OPER_MODE                      |       | VARCHAR2 (9)  |
| PARAMETER                      |       | VARCHAR2 (80) |
| INITIAL_SIZE                   |       | NUMBER        |
| TARGET_SIZE                    |       | NUMBER        |
| FINAL_SIZE                     |       | NUMBER        |
| STATUS                         |       | VARCHAR2 (9)  |

|            |        |
|------------|--------|
| START_TIME | DATE   |
| END_TIME   | DATE   |
| CON_ID     | NUMBER |
| SQL>       |        |

- Display the history of all memory resize operation from v\$memory\_resize\_ops.

```
SQL> column component format a20
SQL> select component, oper_type, initial_size, target_size,
final_size, end_time from v$memory_resize_ops
where OPER_TYPE !='STATIC' order by end_time;
```

| COMPONENT            | OPER_TYPE    | INITIAL_SIZE | TARGET_SIZE | FINAL_SIZE | END_TIME  |
|----------------------|--------------|--------------|-------------|------------|-----------|
| DEFAULT buffer cache | INITIALIZING | 1224736768   | 1224736768  | 1224736768 | 22-OCT-20 |
| streams pool         | GROW         | 16777216     | 33554432    | 33554432   | 22-OCT-20 |
| DEFAULT buffer cache | SHRINK       | 1224736768   | 1207959552  | 1207959552 | 22-OCT-20 |
| shared pool          | GROW         | 637534208    | 654311424   | 654311424  | 22-OCT-20 |
| DEFAULT buffer cache | SHRINK       | 1207959552   | 1191182336  | 1191182336 | 22-OCT-20 |
| shared pool          | GROW         | 654311424    | 671088640   | 671088640  | 22-OCT-20 |
| DEFAULT buffer cache | SHRINK       | 1191182336   | 1174405120  | 1174405120 | 22-OCT-20 |
| DEFAULT buffer cache | SHRINK       | 1174405120   | 1157627904  | 1157627904 | 23-OCT-20 |
| shared pool          | GROW         | 671088640    | 687865856   | 687865856  | 23-OCT-20 |
| large pool           | GROW         | 16777216     | 33554432    | 33554432   | 23-OCT-20 |
| DEFAULT buffer cache | SHRINK       | 1157627904   | 1140850688  | 1140850688 | 23-OCT-20 |
| large pool           | SHRINK       | 33554432     | 16777216    | 16777216   | 23-OCT-20 |
| DEFAULT buffer cache | GROW         | 1140850688   | 1157627904  | 1157627904 | 23-OCT-20 |

13 rows selected.

SQL>

The history shows an increasing demand for space by the shared\_pool (i.e. GROW oper\_type) and the memory used for the growth has been coming from DEFAULT buffer cache (i.e. with the SHRINK oper\_type). If this behavior persists, it is a good indication that either more memory is needed for the database instance or, more likely, a minimum needs to be set on the DEFAULT buffer cache.

- Exit SQL\*plus.

```
SQL> exit
...
$
```

- Close all terminals.

# **Practices for Lesson 33: Analyzing SQL and Optimizing Access Paths**

## **Practices for Lesson 33: Overview**

---

### **Overview**

In these practices, you will use the SQL Tuning Advisor to optimize SQL performance.

## Practice 33-1: 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. Practice 2-1: Using Enterprise Manager Database Express to Manage Performance has been performed and EM Database has been enabled for Flash operation.

### Tasks

#### Initiate a SQL Load

1. Open a new terminal window and source the `oraenv` script.

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

2. Execute the `$HOME/labs/DBMod_MonTune/PERF_setup_tuning.sh` shell script. This script creates a user named `OE`, a tablespace named `TBS_APP`, and a schema named `OE` in the `TBS_APP` tablespace in `ORCLPDB1`. It does the same thing in `ORCLPDB2`. 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.

```
$ $HOME/labs/DBMod_MonTune/PERF_setup_tuning.sh
The Oracle base remains unchanged with value /u01/app/oracle
...
412129 rows created.

Commit complete.
$
```

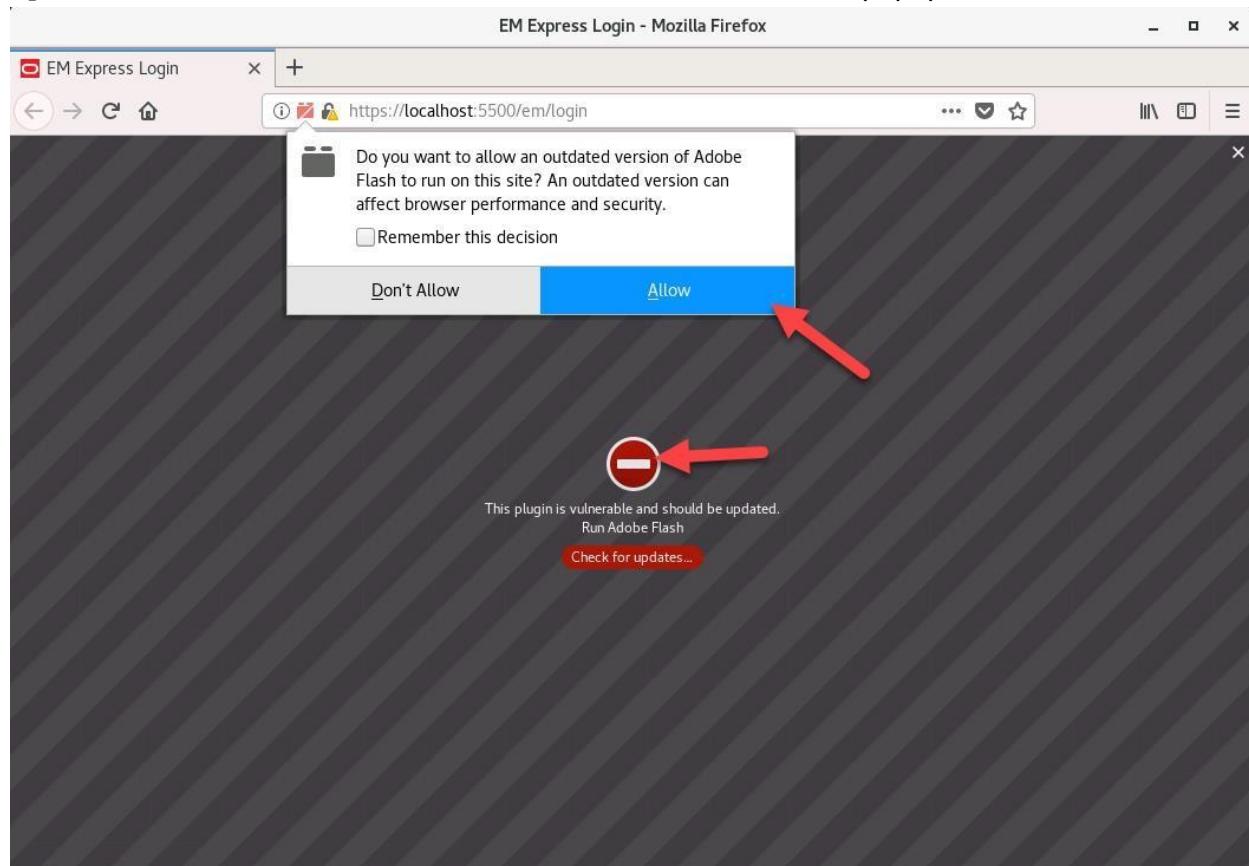
3. Start an application workload in `ORCLPDB1` and `ORCLPDB2`. The `PERF_loop.sh` script runs a SQL script named `PERF_loop.sql` eight times in `ORCLPDB1` as the `OE` user. It then runs the same SQL script eight times in `ORCLPDB2` as the `SYSTEM` user. Let this script run several minutes before moving onto the next step.

```
$ $HOME/labs/DBMod_MonTune/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.

4. Open a browser and launch Enterprise Manager Database Express by entering the following URL: <https://localhost:5500/em> or using the EM Express link on the bookmark toolbar.
5. If a message appears saying: This Pluging is vulnerable and should be updated... Click on the red circle and then **Allow** button on the popup.



6. On the Login page, enter the username **sys** and the password. Leave the Container Name box empty, select **as sysdba**, and click **Login**. See the “Course Practice Environment:

*Security Credentials*" document in your Activity Guide for the password.

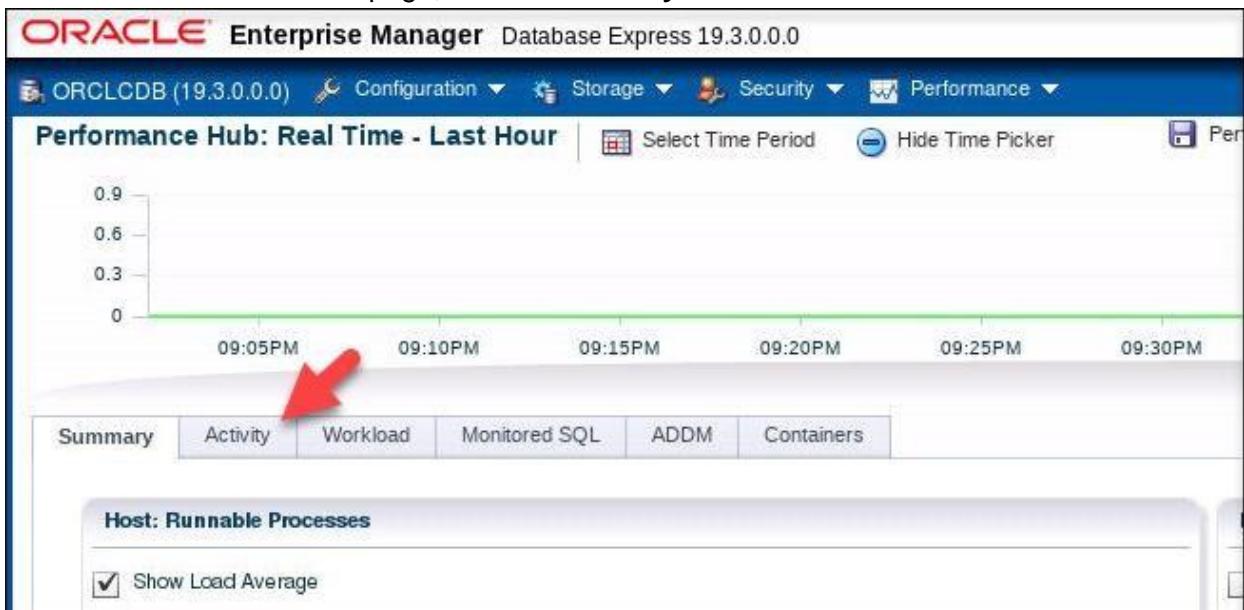


The screenshot shows the Oracle Database Express 19.3.0.0.0 login interface. It has a "Login" title at the top. There are three text input fields: "User Name" containing "sys", "Password" containing "\*\*\*\*\*", and "Container Name" which is empty. Below these is a checked checkbox labeled "as sysdba". At the bottom is a "Login" button.

7. Select **Performance** and then **Performance Hub**.



8. From the Performance Hub page, click the **Activity** tab.



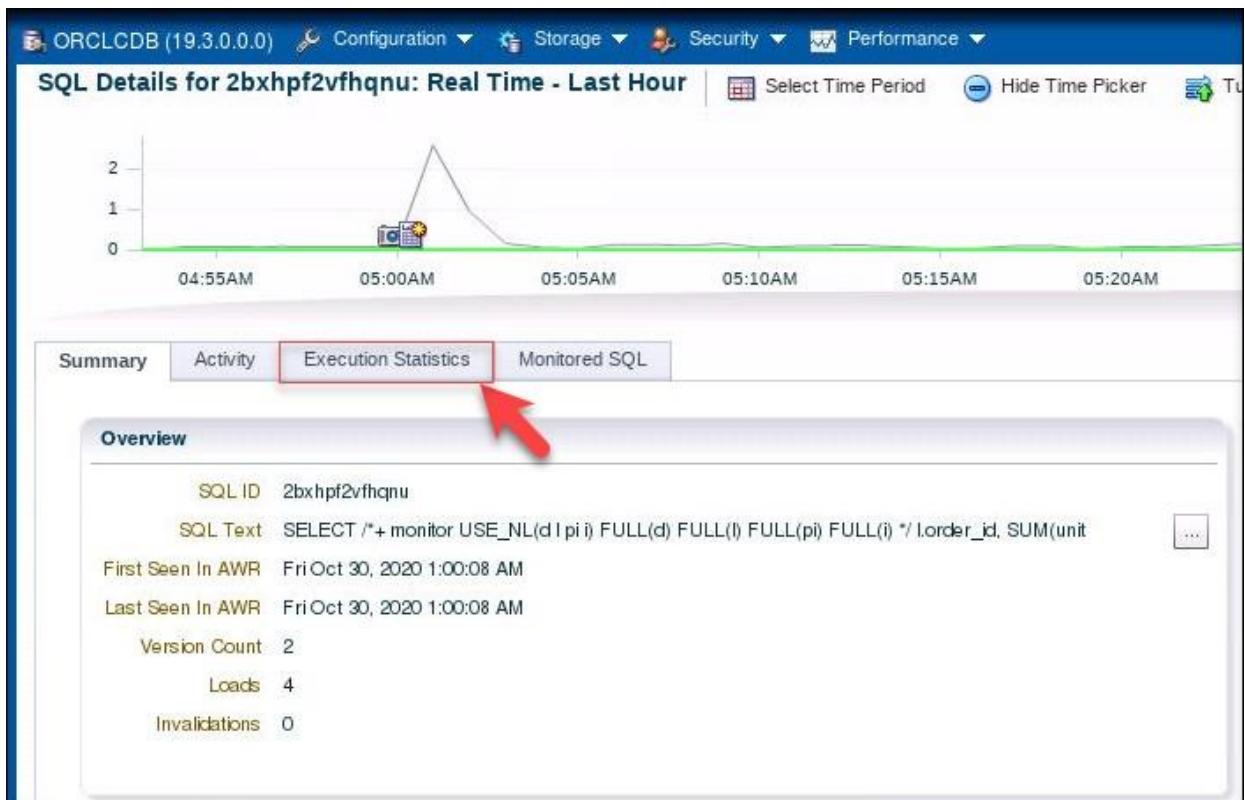
9. The current SQL executions are listed in the table at the bottom of the page. In this example, you can see that there is one consuming SQL execution (the first SQL ID in the list).

| SQL ID        | Activity (Average Active Sessions) |
|---------------|------------------------------------|
| 2bxhpf2vhqnu  | .14                                |
| 0w26sk6t6gc98 | <.01                               |
| 1d25xwrbnv4ub | <.01                               |
| 5dqz0hqt9fru  | <.01                               |

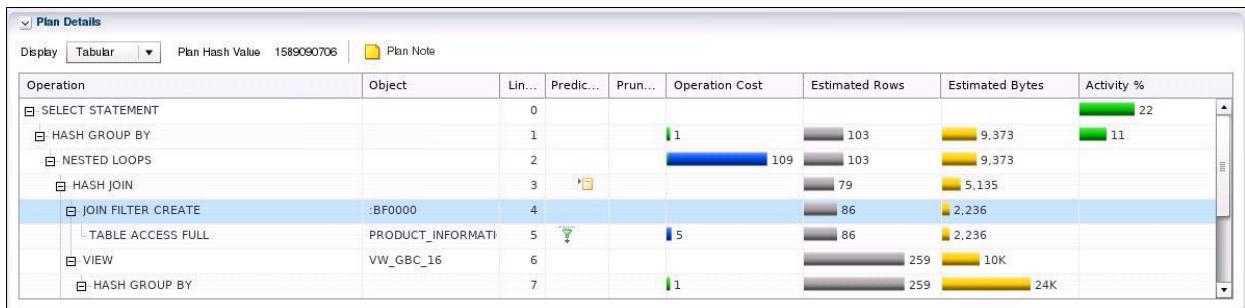
10. Position your cursor over the SQL ID. The following code should appear. If your result looks different, wait a moment and refresh EM Express. Click the SQL ID of the greatest consumer.

| SQL ID       | Activity (Average Active Sessions)                                                     |
|--------------|----------------------------------------------------------------------------------------|
| 2bxhpf2vhqnu | .1                                                                                     |
| 5dqz0hqt9fru | FULL(d)<br>SELECT /*+ monitor USE_NL(d i pi i) FULL(d) FULL(i) FULL(pi) FULL(i) */ ... |
| 9yv5dwv8k0a  |                                                                                        |

11. On SQL Details for ... page, click the **Execution Statistics** tab.



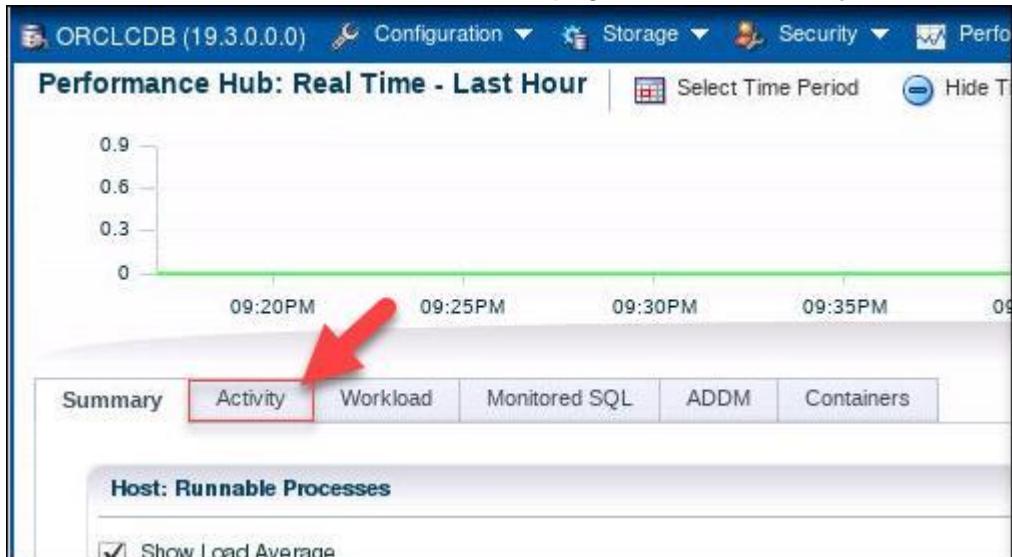
12. The Plan Details area at the bottom of the page shows you the current plan for executing the SQL.



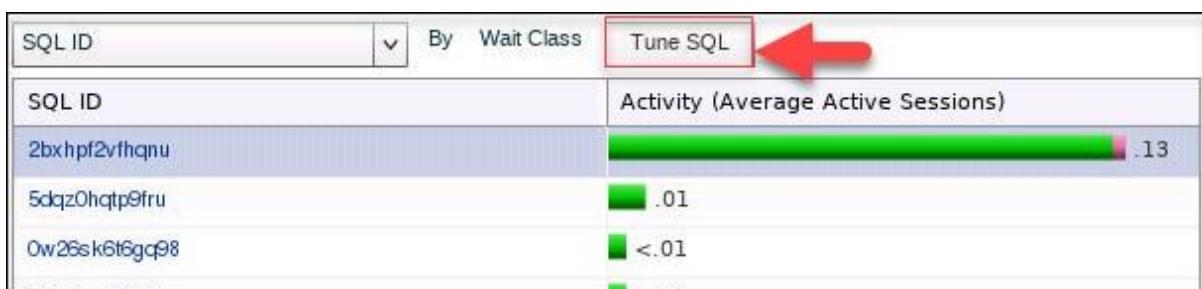
13. To return to where you were last, select **Performance** and then **Performance Hub**.



14. Once back on the main Performance Hub page, click the **Activity** tab.



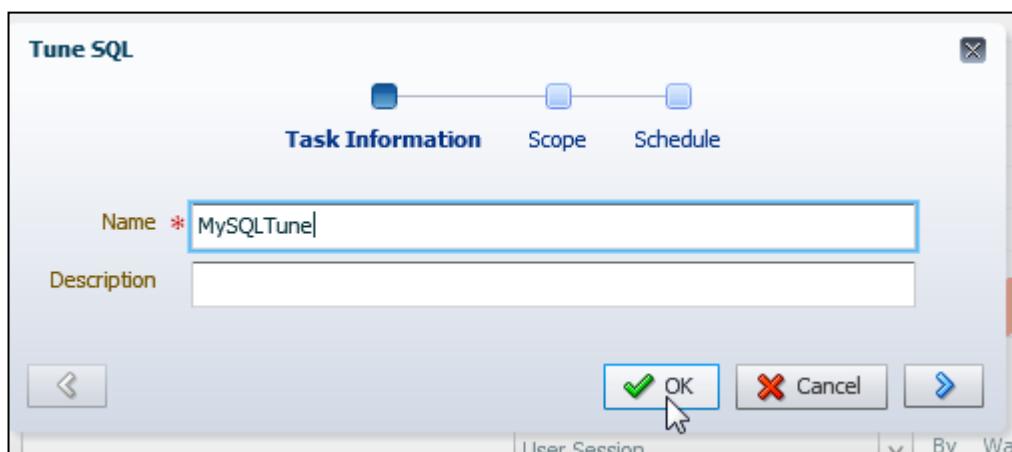
15. In the Activity column, click the row to select the SQL statement to tune and click the **Tune SQL** button to launch the SQL Tuning Advisor.



16. 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.

17. In the Tune SQL dialog box, enter the task name **MySQLTune** and click **OK**.



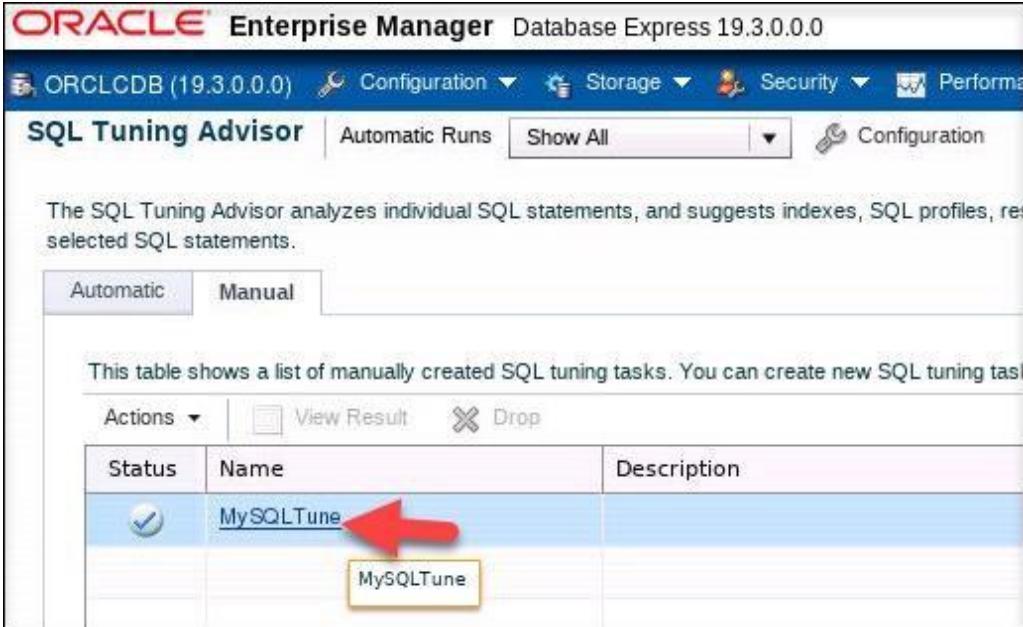
18. While the analysis task is completing, you are automatically brought to the SQL Tuning Advisor page, which has two tabs:

- The Automatic tab lists the automatic tasks executed every night.
- The Manual tab lists the manually created SQL tuning tasks, click the **Manual** tab.

The screenshot shows the Oracle Enterprise Manager Database Express interface. The title bar reads 'ORACLE® Enterprise Manager Database Express 19.3.0.0.0'. The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation bar, the 'SQL Tuning Advisor' tab is selected. A sub-menu bar shows 'Automatic Runs' and 'Show All'. A red arrow points to the 'Manual' tab button, which is highlighted with a red border. A descriptive text box states: 'The SQL Tuning Advisor analyzes individual SQL statements, and suggests indexes, SQL profiles, restructured SQL, selected SQL statements.' On the left, a 'Status' panel displays task details: Task Name: SYS\_AUTO\_SQL\_TUNING\_TASK, Task Owner: SYS, Started: Fri Oct 16, 2020 10:00:02 PM. On the right, a 'Findings' panel shows 'Global Findings' with a count of 100.

19. The task you just created is listed. You may need to wait a moment for it to complete its processing if the status shows the running icon 

If status shows completed icon , then click your MySQLTune to read the recommendations.



The screenshot shows the Oracle Enterprise Manager interface. The title bar reads "ORACLE Enterprise Manager Database Express 19.3.0.0.0". The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation is a sub-menu for "SQL Tuning Advisor" with tabs for "Automatic Runs" and "Show All". A button for "Configuration" is also present. A descriptive text block states: "The SQL Tuning Advisor analyzes individual SQL statements, and suggests indexes, SQL profiles, reselected SQL statements." Below this is a table with two tabs: "Automatic" (selected) and "Manual". The table header includes "Actions", "Status", "Name", and "Description". A red arrow points to the "Status" column of the first row, where a blue checkmark icon is visible next to the task name "MySQLTune".

| Actions                                          | Status                                                                                      | Name      | Description |
|--------------------------------------------------|---------------------------------------------------------------------------------------------|-----------|-------------|
| <a href="#">View Result</a> <a href="#">Drop</a> |  MySQLTune | MySQLTune |             |

20. Examine the Turing Results page for your session/statement and read the recommendations.

**SQL Details**

- Task Name: MySQLTune
- Task Owner: SYS
- SQL ID: 2bxhpf2vhqnu
- Schema: OE
- Container Name: ORCLPDB1

**SQL Text**

```

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

```

**Select Recommendation**

Only one recommendation should be implemented.

| Type        | Findings                                                                                                                                                                                                                                                                        | Type | Benefit (%) |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-------------|
| Statistics  | Optimizer statistics are not up-to-date for objects referenced in the SQL statement.<br>Table "OE"."ORDER_ITEMS" was not analyzed.<br>Table "OE"."INVENTORIES" was not analyzed.<br>Table "OE"."PRODUCT_INFORMATION" was not analyzed.<br>Table "OE"."ORDERS" was not analyzed. |      |             |
| SQL Profile | A potentially better execution plan was found for this statement.                                                                                                                                                                                                               |      | 96.9        |

21. In the SQL Text area, you can view the full SQL statement to be tuned. Scroll down to view all of it.

**SQL Text**

```

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

```

22. 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, which are not analyzed.

The screenshot shows a 'Select Recommendation' interface. At the top, it says 'Only one recommendation should be implemented.' Below are two rows in a table:

| Type        | Findings                                                                                                                                                                                                                                                                        | Benefit (%) |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Statistics  | Optimizer statistics are not up-to-date for objects referenced in the SQL statement.<br>Table "OE"."ORDER_ITEMS" was not analyzed.<br>Table "OE"."INVENTORIES" was not analyzed.<br>Table "OE"."PRODUCT_INFORMATION" was not analyzed.<br>Table "OE"."ORDERS" was not analyzed. |             |
| SQL Profile | A potentially better execution plan was found for this statement.                                                                                                                                                                                                               | 96.9        |

23. 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.

24. In the **Select Recommendations** area, click the **Statistics** link.

The screenshot shows the same 'Select Recommendation' interface as before, but the 'Statistics' link in the table header is highlighted with a red arrow. The table data remains the same as in the previous screenshot.

| Type       | Findings                                                                                                                                                                                                                                                                        |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Statistics | Optimizer statistics are not up-to-date for objects referenced in the SQL statement.<br>Table "OE"."ORDER_ITEMS" was not analyzed.<br>Table "OE"."INVENTORIES" was not analyzed.<br>Table "OE"."PRODUCT_INFORMATION" was not analyzed.<br>Table "OE"."ORDERS" was not analyzed. |

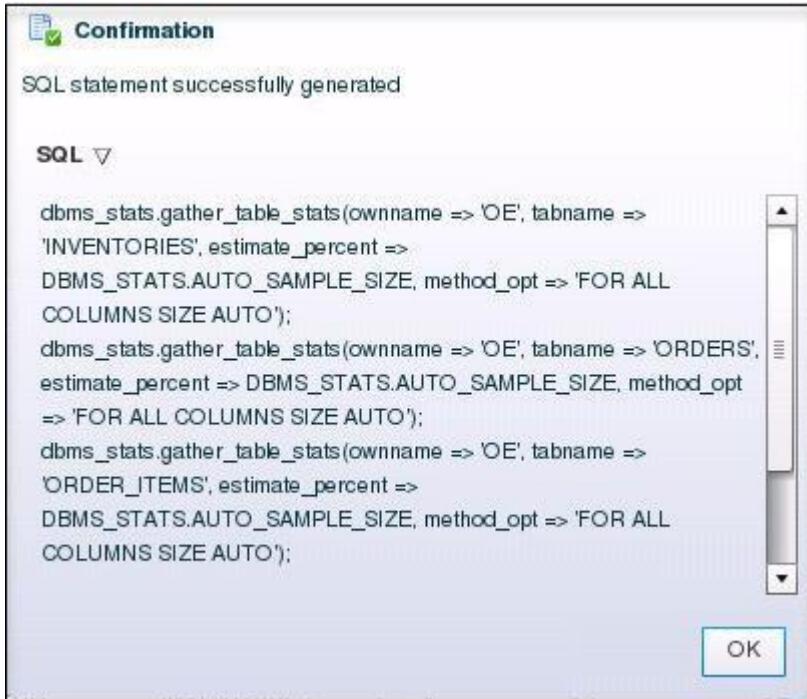
25. On the Recommendation Details page, view the list of recommendations and then click the **Implement** button at the top of the page. Notice that the Optimizer recommends gathering statistics for the tables in your SQL statement.

The screenshot shows the 'Recommendation Details' page in Oracle SQL Developer. At the top, there is a toolbar with buttons for 'Implement' (highlighted by a red arrow) and 'Validate with SPA'. The status bar indicates 'Page Refreshed 10:36:31 PM GMT'. Below the toolbar, there is a section titled 'Stale or Missing Statistics' which contains a message about the optimizer relying on object statistics. Under 'Recommendation(s)', there is a list of four tables that were not analyzed: 'PRODUCT\_INFORMATION', 'INVENTORIES', 'ORDER\_ITEMS', and 'ORDERS'. Below this, there is a section titled 'Compare Explain Plans' with a table showing the execution plan for a 'SELECT STATEMENT'. The table has columns for Operation, Object, L..., Pre..., Pr..., Operation C..., Estimated R..., and Estimated B... . The first row shows 'SELECT STATEMENT' with values 0, 453, and 53K respectively.

26. In the Gather Statistics dialog box, click **Show SQL**.



27. View the SQL package and procedure that would gather the statistics and click **OK**.



The screenshot shows a confirmation dialog box titled "Confirmation". It displays a SQL statement that has been successfully generated. The SQL code is as follows:

```
SQL statement successfully generated

SQL ▼

dbms_stats.gather_table_stats(ownname => 'OE', tablename =>
'INVENTORIES', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL
COLUMNS SIZE AUTO');
dbms_stats.gather_table_stats(ownname => 'OE', tablename => 'ORDERS',
estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt
=> 'FOR ALL COLUMNS SIZE AUTO');
dbms_stats.gather_table_stats(ownname => 'OE', tablename =>
'ORDER_ITEMS', estimate_percent =>
DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR ALL
COLUMNS SIZE AUTO');

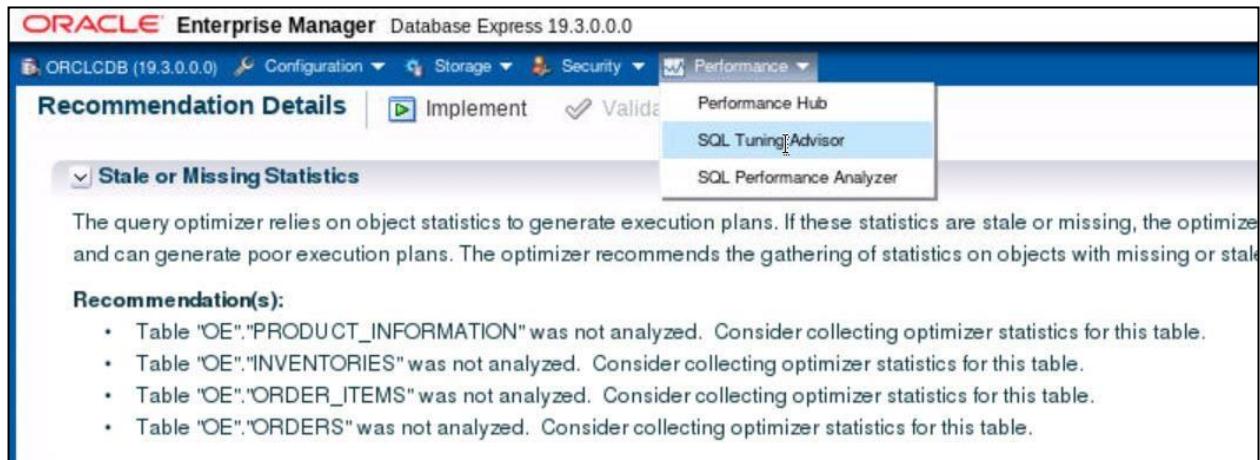
OK
```

28. 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.

29. Select **Performance** and then **SQL Tuning Advisor**.



The screenshot shows the Oracle Enterprise Manager Database Express interface. The top navigation bar is blue, and the main menu includes "ORCLCDB (19.3.0.0.0)", "Configuration", "Storage", "Security", and "Performance". The "Performance" menu is open, showing options: "Performance Hub", "SQL Tuning Advisor" (which is highlighted in blue), and "SQL Performance Analyzer". Below the menu, there is a section titled "Recommendation Details" with tabs "Implement" and "Validate". A checkbox "Stale or Missing Statistics" is checked. A message states: "The query optimizer relies on object statistics to generate execution plans. If these statistics are stale or missing, the optimizer can generate poor execution plans. The optimizer recommends the gathering of statistics on objects with missing or stale statistics." A section titled "Recommendation(s):" lists four tables that were not analyzed: "OE"."PRODUCT\_INFORMATION", "OE"."INVENTORIES", "OE"."ORDER\_ITEMS", and "OE"."ORDERS". Each entry includes a bullet point and a descriptive sentence about collecting optimizer statistics for the table.

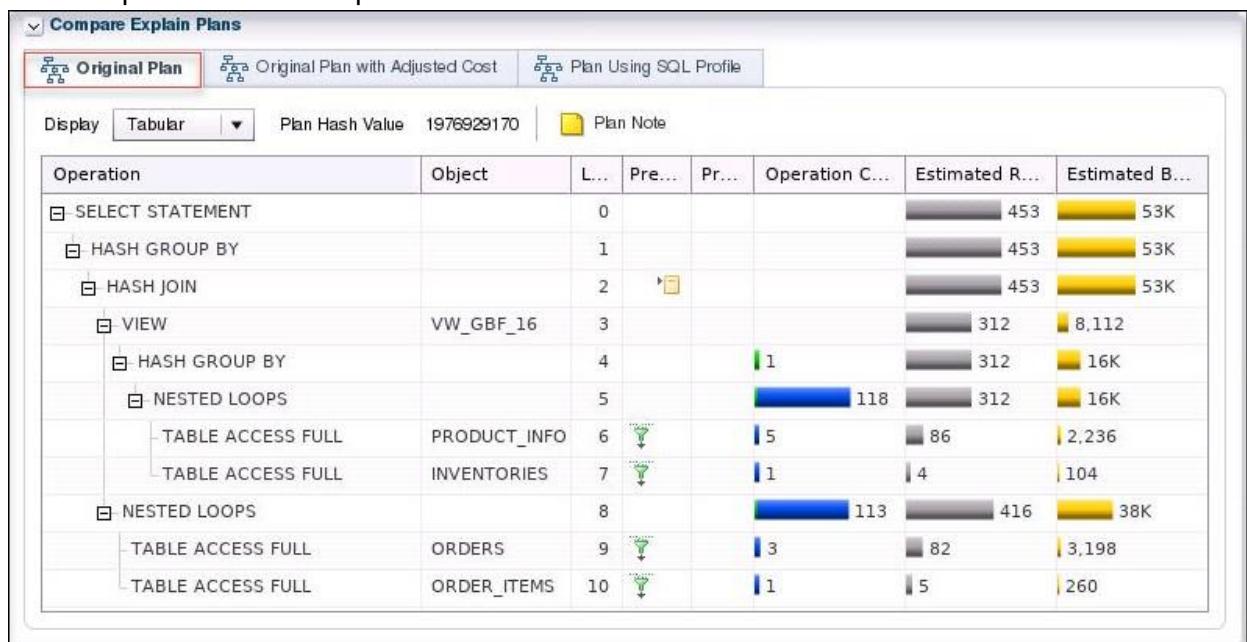
30. Click the **Manual** tab, if needed.  
31. Click **MySQLTune** (or the name you gave your tuning task).

32. 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. In the Select Recommendation section, click the SQL Profile link.

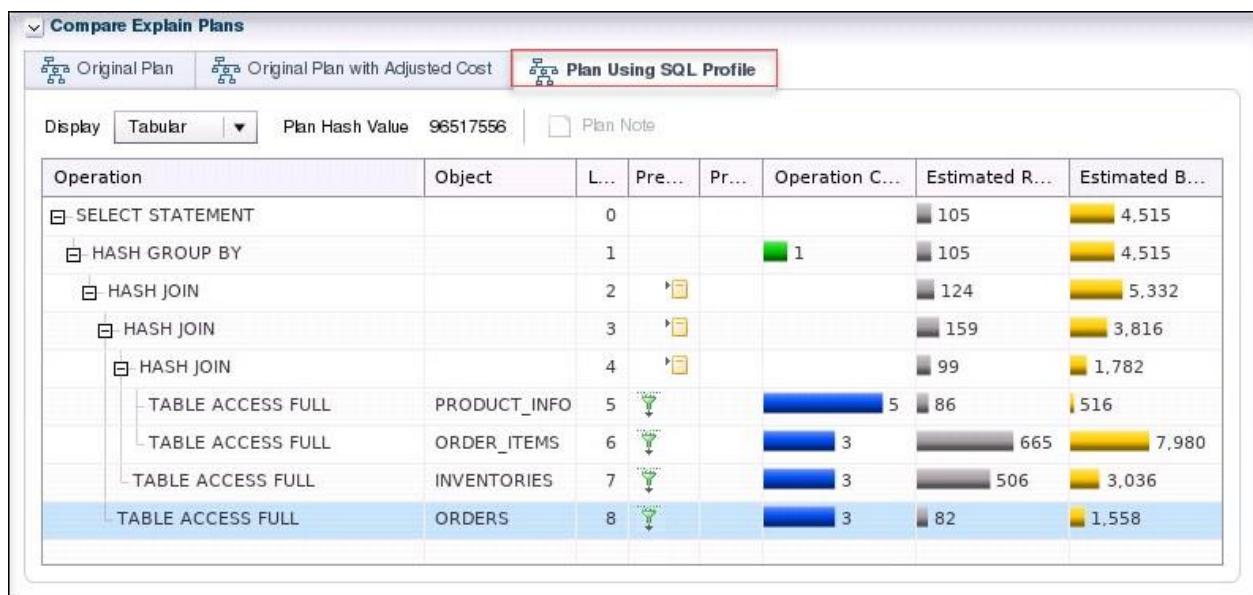
The screenshot shows a table with columns 'Type', 'Findings', and 'Benefit (%)'. The 'Findings' column contains text about optimizer statistics for tables like ORDER\_ITEMS, INVENTORIES, PRODUCT\_INFORMATION, and ORDERS. The 'Benefit (%)' column shows a progress bar and the value '96.9'. A red arrow points to the 'SQL Profile' row.

| Type        | Findings                                                                        | Benefit (%)                                                            |
|-------------|---------------------------------------------------------------------------------|------------------------------------------------------------------------|
| Statistics  | Optimizer statistics are not up-to-date for objects referenced in the SQL st... |                                                                        |
| SQL Profile | A potentially better execution plan was found for this statement.               | <div style="width: 96.9%;"><div style="width: 96.9%;">96.9</div></div> |

33. On the Original Plan tab, which is displayed by default, view the SQL execution plan. You saw this plan earlier in the practice.



34. Click the Plan Using SQL Profile tab and view its SQL execution plan. Notice the differences between it and the original plan.



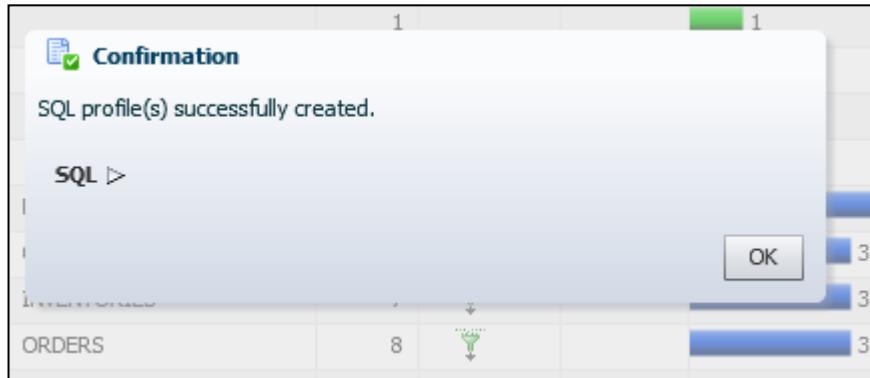
35. At the top of the page, click the **Implement** button
36. In the Create SQL Profile dialog box, click **Show SQL**.
37. A Confirmation dialog box shows you the generated SQL statement. Click **OK**.



38. In the Create SQL Profile dialog box, click **OK** to implement the new profile.



39. The SQL profile is created. In the Confirmation dialog box, click **OK**.



40. In the terminal window, if your workload script is still running, press **Ctrl+c** to stop the activity.

### 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.

41. Return to the terminal window.. Set Start SQL\*Plus and connect to the CDB root as the `SYS` user with the `SYSDBA` privilege.

```
$. oraenv
ORACLE_SID = [orclcldb] ? orclcldb
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus / as sysdba
...
SQL>
```

42. 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>
```

43. Remove any blocks of the tables (selected in the query) from the buffer cache.

```
SQL> alter system flush buffer_cache;

System altered.

SQL>
```

44. Exit SQL\*Plus.

```
SQL> exit
...
$
```

45. Run the application workload again in ORCLPDB1 and ORCLPDB2. The `PERF_loop.sh` script runs a SQL script named `PERF_loop.sql` eight times in ORCLPDB1 as the `OE` user. It then runs the same SQL script eight times in ORCLPDB2 as the `SYSTEM` user.

```
$ $HOME/labs/DBMod_MonTune/PERF_loop.sh
The Oracle base remains unchanged with value /u01/app/oracle
...
```

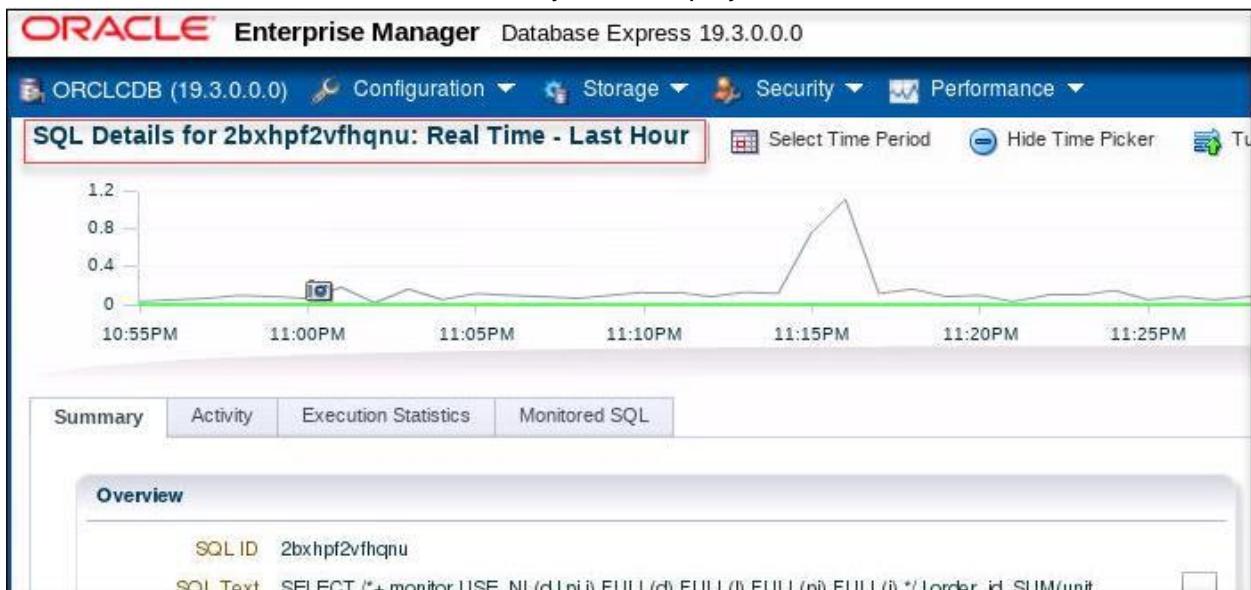
46. Wait at least 3 minutes then return to EM Express.  
47. If a Warning dialog box is displayed stating that a particular SQL ID is no longer from Cursor Cache, click **OK**.  
48. Select **Performance** and then **Performance Hub**.  
49. Click the **Activity** tab.  
50. At the bottom of the page, note the value in the Activity column.



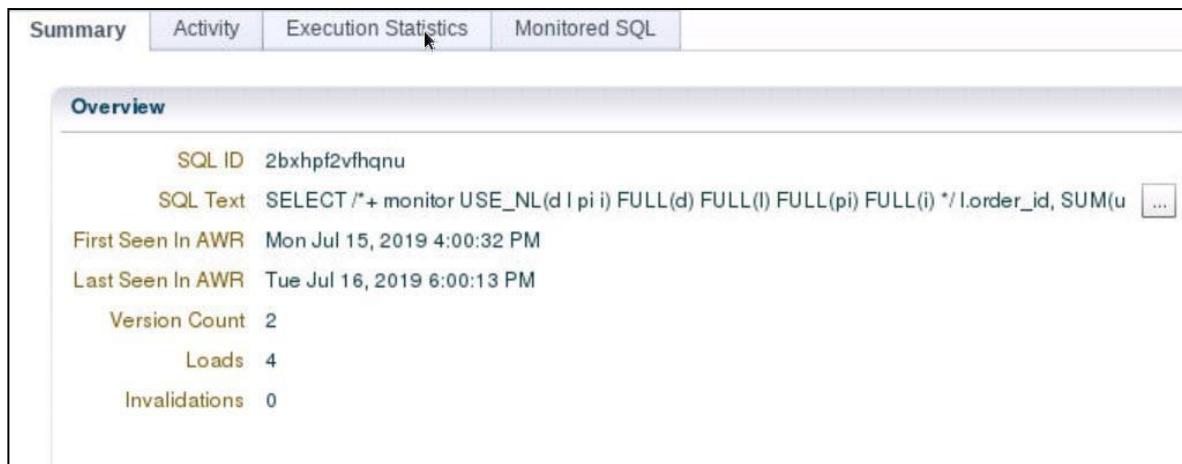
51. Question: How does the value in the Activity column now compare to the value in the Activity column prior to SQL tuning? Is there a performance benefit to the SQL tuning that you just did?

Answer: In this example, the average active sessions value went down from .11 to .01, so yes, there is a performance benefit from tuning the SQL. Your values may differ.

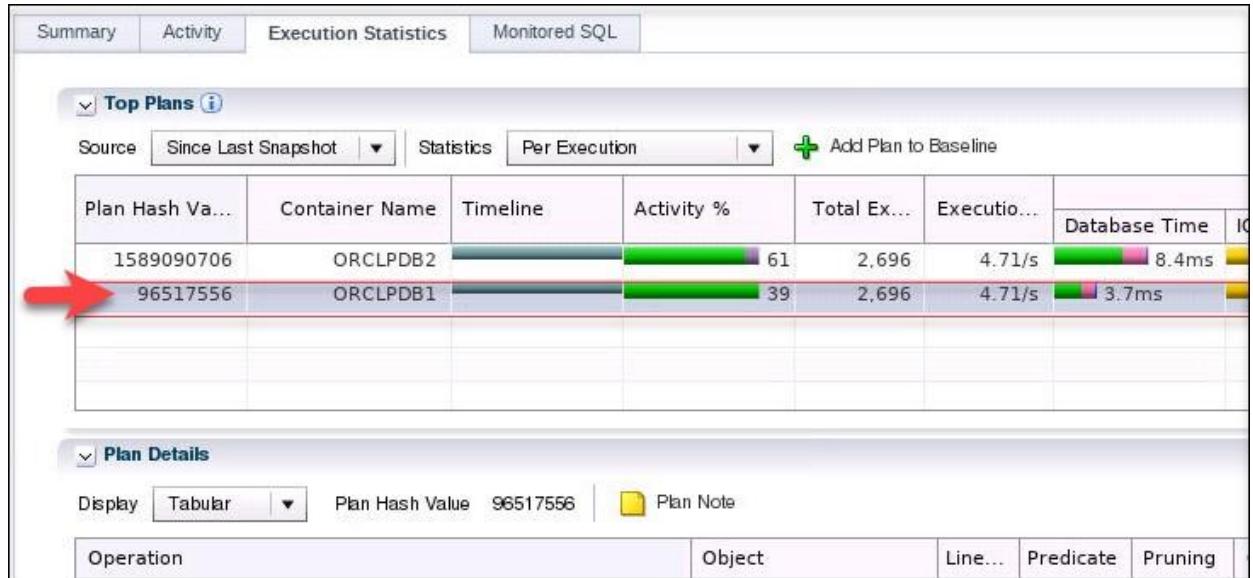
52. Click the link for the SQL ID. The Summary tab is displayed for the SQL ID.



53. Click the **Execution Statistics** tab.



54. In the Top Plans area, ensure the row with the Container Name ORCLPDB1 is selected.



55. In the Plan Details area, notice that the plan now used is the plan that uses the SQL Profile (refer back to **step 6** in the previous section in this practice).



56. Click **Log Out** to exit EM Express and close the browser window.

57. In the terminal window, press **Ctrl+C** to stop the activity.

## Practice 33-2: 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:

- In the terminal window's menu, select **Terminal** and then **Set Title**. A Set Title dialog box is displayed.
- In the Title box, enter the name “**Window**” and a number
- Click **OK**.

### Assumptions

You are logged in as the `oracle` user.

### Tasks

#### Window 1: Start an Application Workload

1. In the open terminal, name it Window 1, then execute the `$HOME/labs/DBMod_MonTune/PERF_setup_tuning.sh` shell script. Wait for the setup script to finish. You can ignore any error messages because they are expected.

```
$ $HOME/labs/DBMod_MonTune/PERF_setup_tuning.sh
...
412129 rows created.

Commit complete.

$
```

2. Start an application workload in ORCLPDB1 and ORCLPDB2.

```
$ $HOME/labs/DBMod_MonTune/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.

3. Open another new terminal window. This window will be referred to as Window 2.
4. Start SQL\*Plus and connect to ORCLPDB1 as the SYS user. Refer to *Practice Environment: Security Credentials* for the password value.

```
$ sqlplus sys/password@orclpdb1 as sysdba
...
SQL>
```

5. Execute the \$HOME/labs/DBMod\_MonTune/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 the previous practice, 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/DBMod_MonTune/OPTADV_1.sql
```

```
Procedure created.
```

```
SQL>
```

6. Create and execute an advisor task named my\_task by executing the \$HOME/labs/DBMod\_MonTune/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_implementations_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/DBMod_MonTune/OPTADV_2.sql
```

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

```
SQL>
```

7. Verify that the procedure completed successfully.

- a. Query the `USER_ADVISOR_TASKS` view.

```

SQL> column advisor_name format a20
SQL> column execution_type format a15
SQL> column last_execution format a15
SQL> column status format a15
SQL> select advisor_name, execution_type, last_execution, status
from user_advisor_tasks where task_name = 'MY_TASK';

ADVISOR_NAME EXECUTION_TYPE LAST_EXECUTION STATUS
----- -----
Statistics Advisor STATISTICS EXEC_165 COMPLETED
SQL>

```

- b. Query the `USER_ADVISOR_EXECUTIONS` view. The results below are formatted for easier viewing. Your dates and numbers of rows 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_165`.

```

SQL> column task_name format a25
SQL> column execution_name format a12
SQL> column type format a12
SQL> column status format a12
SQL> select task_name, execution_name, execution_end,
execution_type as type, status from user_advisor_executions
order by 3;

TASK_NAME EXECUTION_NAME EXECUTION_TYPE STATUS
----- -----
... ...
SYS_AUTO_SPM_EVOLVE_TASK EXEC_143 21-OCT-20 SPM EVOLVE COMPLETED

```

|                          |          |           |            |           |
|--------------------------|----------|-----------|------------|-----------|
| AUTO_STATS_ADVISOR_TASK  | EXEC_144 | 21-OCT-20 | STATISTICS | COMPLETED |
| SYS_AUTO_SPM_EVOLVE_TASK | EXEC_153 | 22-OCT-20 | SPM EVOLVE | COMPLETED |
| AUTO_STATS_ADVISOR_TASK  | EXEC_154 | 22-OCT-20 | STATISTICS | COMPLETED |
| MY_TASK                  | EXEC_165 | 23-OCT-20 | STATISTICS | COMPLETED |

29 rows selected.

SQL>

## 8. Generate a report.

- Execute the \$HOME/labs/DBMod\_MonTune/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/DBMod\_MonTune/OPTADV\_3.sql

PL/SQL procedure successfully completed.

SQL>

- Execute the \$HOME/labs/DBMod\_MonTune/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/DBMod_MonTune/OPTADV_4.sql
```

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

```
SQL>
```

9. Edit the script \$HOME/labs/DBMod\_MonTune/OPTADV\_5.sql with vi. Be sure to edit the script and enter the correct value for F.EXECUTION\_NAME as determined in step 5b.

```
SQL> ! vi $HOME/labs/DBMod_MonTune/OPTADV_5.sql
...
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_165';
```

10. Execute the \$HOME/labs/DBMod\_MonTune/OPTADV\_5.sql script to view the findings.

```
SQL> @$HOME/labs/DBMod_MonTune/OPTADV_5.sql
```

```
FINDING_ID MESSAGE
```

```
BENEFIT_TYPE
```

```
1 There are 1 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.
Set the CONCURRENT preference.
```

```
1 There are 1 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.
```

11. Generate the script before a possible implementation.

- a. Execute the \$HOME/labs/DBMod\_MonTune/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/DBMod_MonTune/OPTADV_6.sql
```

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

```
SQL>
```

- b. Execute the \$HOME/labs/DBMod\_MonTune/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/DBMod_MonTune/OPTADV_7.sql
```

```
-- Script generated for the recommendations from execution
EXEC_165
-- in the statistics advisor task MY_TASK
-- Script version
12.2
```

```
-- No scripts will be provided for the rule USEAUTOJOB. Please
check the report for more details.

-- No scripts will be
provided for the rule COMPLETEAUTOJOB. Please check the report
for more details.

-- No scripts will be provided for the rule
MAINTAINSTATSHISTORY. Please check the report for more details.

-- No scripts will be provided for the rule
TURNONSQLPLANDIRECTIVE. Please check the report for more
details.

-- No scripts will be provided for the rule
AVOIDSETPROCEDURES. Please check the report for more details.

-- No scripts will be provided for the rule
USEDEFAULTPARAMS. Please
check the report for more details.

-- No scripts will be provided for the rule
USEGATHERSCHEMASTATS. Please check the report for more
details.

-- No scripts will be provided for the rule
AVOIDINEFFICIENTSTATSOPRSEQ. Please check the report for more
details.

-- No
scripts will be provided for the rule
AVOIDUNNECESSARYSTATSCOLLECTION. Please check the report for
more details.

-- No scripts will
be provided for the rule GATHERSTATSAFTERBULKDML. Please check
the report for more details.

-- No scripts will be provided for the
rule AVOIDDROPRECREATE. Please check the report for more
details.

-- No scripts will be provided for the rule
AVOIDOUTOFRANGE. Please
check the report for more details.

-- No scripts will be provided for the rule
AVOIDANALYZETABLE. Please check the report for more
details.

-- No scripts will be provided for the rule USEAUTOJOB. Please
check the report for more details.

-- No scripts will be
provided for the rule COMPLETEAUTOJOB. Please check the report
for more details.

-- No scripts will be provided for the rule
```

```
MAINTAINSTATSHISTORY.Please check the report for more details.
-- No scripts will be provided for the rule
TURNONSQLPLANDIRECTIVE.Please check the report for more
details.
-- No scripts will be provided for the rule
AVOIDSETPROCEDURES.Please check the report for more details.
-- No scripts will be provided for the rule
USEDEFAULTPARAMS.Please
check the report for more details.
-- No scripts will be provided for the rule
USEGATHERSCHEMASTATS.Please check the report for more
details.
-- No scripts will be provided for the rule
AVOIDINEFFICIENTSTATSOPRSEQ.Please check the report for more
details.
-- No
scripts will be provided for the rule
AVOIDUNNECESSARYSTATSCOLLECTION.Please check the report for
more details.
-- No scripts will
be provided for the rule GATHERSTATSAFTERBULKDML.Please check
the report for more details.
-- No scripts will be provided for the
rule AVOIDDROPRECREATE.Please check the report for more
details.
-- No scripts will be provided for the rule
AVOIDOUTOFRANGE.Please
check the report for more details.
-- No scripts will be provided for the rule
AVOIDANALYZETABLE.Please check the report for more
details.

-- Scripts for rule USECONCURRENT
-- Rule Description: Use Concurrent preference for Statistics
Collection

-- No
scripts will be provided for the rule USEAUTOJOB.Please check
the report for more details.
-- No scripts will be provided for the
rule COMPLETEAUTOJOB.Please check the report for more details.
-- No scripts will be provided for the rule
MAINTAINSTATSHISTORY.Please check the report for more details.
```

```

-- No scripts will be provided for the rule
TURNONSQLPLANDIRECTIVE.Please check the report for more
details.

-- No scripts will be provided for the rule
AVOIDSETPROCEDURES.Please check the report for more details.

-- No scripts will be provided for the rule
USEDEFAULTPARAMS.Please
check the report for more details.

-- No scripts will be provided for the rule
USEGATHERSCHEMASTATS.Please check the report for more
details.

-- No scripts will be provided for the rule
AVOIDINEFFICIENTSTATSOPRSEQ.Please check the report for more
details.

-- No
scripts will be provided for the rule
AVOIDUNNECESSARYSTATSCOLLECTION.Please check the report for
more details.

-- No scripts will
be provided for the rule GATHERSTATSAFTERBULKDM.LPlease check
the report for more details.

-- No scripts will be provided for the
rule AVOIDDROPRECREATE.Please check the report for more
details.

-- No scripts will be provided for the rule
AVOIDOUTOFRANGE.Please
check the report for more details.

-- No scripts will be provided for the rule
AVOIDANALYZETABLE.Please check the report for more
details.

-- Scripts for rule USEDEFAULTPREFERENCE
-- Rule Description: Use Default Preference for Stats
Collection
-- Set global
preferenes to default values.

-- Scripts for rule UNLOCKNONVOLATILETABLE
-- Rule Description: Statistics for objects with
non-volatile should not be locked
-- Unlock statistics for objects that are not volatile.

```

```
-- Scripts for rule USEAUTODEGREE
--
Rule Description: Use Auto Degree for statistics collection
-- Turn on auto degree for those objects for which using auto
degree is
helpful.

-- Scripts for rule LOCKVOLATILETABLE
-- Rule Description: Statistics for objects with volatile data
should be locked
--
Lock statistics for volatile objects.

-- Scripts for rule NOTUSEINCREMENTAL
-- Rule Description: Statistics should not be
maintained incrementally when it is not beneficial
-- Turn off incremental option for those objects for which
using incremental is
not helpful.

-- Scripts for rule USEINCREMENTAL
-- Rule Description: Statistics should be maintained
incrementally when it is
beneficial
-- Turn on the incremental option for those objects for which
using incremental is helpful.

-- Scripts for rule
USEDEFAULTOBJECTPREFERENCE
-- Rule Description: Use Default Object Preference for
statistics collection
-- Setting object-level
preferences to default values
-- setting CASCADE to default value for object level
preference
-- setting ESTIMATE_PERCENT to default
value for object level preference
-- setting METHOD_OPT to default value for object level
preference
-- setting GRANULARITY to
```

```

default value for object level preference
-- setting NO_INVALIDATE to default value for object level
preference

-- Scripts for
rule AVOIDSTALEMENTS
-- Rule Description: Avoid objects with stale or no statistics
-- Gather statistics for those objects that are
missing or have no statistics.
-- Scripts for rule MAINTAINSTATSCONSISTENCY
-- Rule Description: Statistics of dependent objects
should be consistent
-- Gather statistics for those objects 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.

12. Question: What does `obj_filter_list` indicate?

Answer: The object filter list contains the names of the ten objects with no statistics.

13. 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.

14. Execute the `$HOME/labs/DBMod_MonTune/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/DBMod_MonTune/OPTADV_8.sql
```

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

15. 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 from dba_tables where owner='OE';
```

| TABLE_NAME          | NUM_ROWS | BLOCKS | EMPTY_BLOCKS | AVG_ROW_LEN | LAST_ANAL |
|---------------------|----------|--------|--------------|-------------|-----------|
| ORDERS              | 105      | 5      | 0            | 38          | 23-OCT-20 |
| ORDER_ITEMS         | 665      | 5      | 0            | 18          | 23-OCT-20 |
| CUSTOMERS           | 319      | 13     | 0            | 158         | 23-OCT-20 |
| INVENTORIES         | 636      | 5      | 0            | 11          | 23-OCT-20 |
| PRODUCT_INFORMATION | 287      | 13     | 0            | 216         | 23-OCT-20 |
| LINEORDER           | 3297032  | 47229  | 0            | 98          | 23-OCT-20 |
| PART                | 1600000  | 20645  | 0            | 87          | 23-OCT-20 |
| SUPPLIER            | 16000    | 260    | 0            | 102         | 23-OCT-20 |
| CUSTOMER            | 240000   | 3846   | 0            | 107         | 23-OCT-20 |
| DATE_DIM            | 2556     | 43     | 0            | 100         | 23-OCT-20 |

```
10 rows selected.
```

16. Drop the advisor task.

```
SQL> exec DBMS_STATS.DROP_ADVISOR_TASK('MY_TASK')
```

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

```
SQL>
```

17. Exit SQL\*Plus and close Window 2.

```
SQL> exit
...
$ exit
```

### **Window 1: Stop the Workload**

18. In Window 1, press Ctrl+c to stop the activity and then close the window.