Oracle Database 19c: Administration

# Table of Contents

[Course Practice Environment: Security Credentials 7](#_TOC_250121)

[Course Practice Environment: Security Credentials 8](#_TOC_250120)

[Practices for Lesson 1: Introduction to Oracle Database 11](#_TOC_250119)

[Practices for Lesson 1 12](#_TOC_250118)

[Practices for Lesson 2: Accessing an Oracle Database 13](#_TOC_250117)

[Practices for Lesson 2 14](#_TOC_250116)

[Practices for Lesson 3: Creating an Oracle Database by Using DBCA 15](#_TOC_250115)

[Practices for Lesson 3: Overview 16](#_TOC_250114)

[Practice 3-1: Creating a New CDB 17](#_TOC_250113)

[Practices for Lesson 4: Creating an Oracle Database by Using a SQL Command 23](#_TOC_250112)

[Practices for Lesson 4: Overview 24](#_TOC_250111)

[Practice 4-1: Creating a New CDB 25](#_TOC_250110)

[Practices for Lesson 5: Starting Up and Shutting Down an Oracle Database 31](#_TOC_250109)

[Practices for Lesson 5: Overview 32](#_TOC_250108)

[Practice 5-1: Shutting Down and Starting Up the Oracle Database 33](#_TOC_250107)

[Practices for Lesson 6: Managing Database Instances 41](#_TOC_250106)

[Practices for Lesson 6: Overview 42](#_TOC_250105)

[Practice 6-1: Investigating Initialization Parameter Files 43](#_TOC_250104)

[Practice 6-2: Viewing Initialization Parameters by Using SQL\*Plus 50](#_TOC_250103)

[Practice 6-3: Modifying Initialization Parameters by Using SQL\*Plus 64](#_TOC_250102)

[Practice 6-4: Viewing Diagnostic Information 73](#_TOC_250101)

[Practices for Lesson 7: Oracle Net Services Overview 81](#_TOC_250100)

[Practices for Lesson 7 82](#_TOC_250099)

[Practices for Lesson 8: Configuring Naming Methods 83](#_TOC_250098)

[Practices for Lesson 8: Overview 84](#_TOC_250097)

[Practice 8-1: Configuring the Oracle Network to Access a Database 85](#_TOC_250096)

[Practice 8-2: Creating a Net Service Name for a PDB 89](#_TOC_250095)

[Practices for Lesson 9: Configuring and Administering the Listener 93](#_TOC_250094)

[Practices for Lesson 9: Overview 94](#_TOC_250093)

[Practice 9-1: Exploring the Default Listener 95](#_TOC_250092)

[Practice 9-2: Creating a Second Listener 107](#_TOC_250091)

[Practice 9-3: Connecting to a Database Service Using the New Listener 115](#_TOC_250090)

[Practices for Lesson 10: Configuring a Shared Server Architecture 117](#_TOC_250089)

[Practices for Lesson 10: Overview 118](#_TOC_250088)

[Practice 10-1: Configuring Shared Server Mode 119](#_TOC_250087)

[Practice 10-2: Configuring Clients to Use a Shared Server 122](#_TOC_250086)

Practices for Lesson 11: Configuring Oracle Connection Manager for Multiplexing and

Access Control 125

[Practices for Lesson 11: Overview 126](#_TOC_250085)

[Practices for Lesson 12: Creating PDBs from Seed 145](#_TOC_250079)

[Practices for Lesson 12: Overview 146](#_TOC_250078)

[Practice 12-1: Creating a New PDB from the PDB Seed 147](#_TOC_250077)

[Practices for Lesson 13: Using Other Techniques to Create PDBs 151](#_TOC_250076)

[Practices for Lesson 13: Overview 152](#_TOC_250075)

[Practice 13-1: Cloning Remote PDBs in Hot Mode 153](#_TOC_250074)

[Practice 13-2: Relocating PDBs 160](#_TOC_250073)

[Practices for Lesson 14: Managing PDBs 169](#_TOC_250072)

[Practices for Lesson 14: Overview 170](#_TOC_250071)

[Practice 14-1: Renaming a PDB 171](#_TOC_250070)

[Practice 14-2: Setting Parameter Values for PDBs 174](#_TOC_250069)

[Practices for Lesson 15: Database Storage Overview 179](#_TOC_250068)

[Practices for Lesson 15: Overview 180](#_TOC_250067)

[Practices for Lesson 16: Creating and Managing Tablespaces 181](#_TOC_250066)

[Practices for Lesson 16: Overview 182](#_TOC_250065)

[Practice 16-1: Viewing Tablespace Information 183](#_TOC_250064)

[Practice 16-2: Creating a Tablespace 192](#_TOC_250063)

[Practice 16-3: Managing Temporary and Permanent Tablespaces 200](#_TOC_250062)

[Practices for Lesson 17: Improving Space Usage 207](#_TOC_250061)

[Practices for Lesson 17: Overview 208](#_TOC_250060)

[Practice 17-1: Managing Space in Tablespaces 209](#_TOC_250059)

[Practice 17-2: Using Compression 220](#_TOC_250058)

[Practice 17-3: Enabling the Resumable Space Allocation Feature 228](#_TOC_250057)

[Practices for Lesson 18: Managing Undo Data 235](#_TOC_250056)

[Practices for Lesson 18: Overview 236](#_TOC_250055)

[Practice 18-1: Managing Undo Tablespaces in a PDB 237](#_TOC_250054)

[Practices for Lesson 19: Creating and Managing User Accounts 239](#_TOC_250053)

[Practices for Lesson 19: Overview 240](#_TOC_250052)

[Practice 19-1: Creating Common and Local Users 241](#_TOC_250051)

[Practice 19-2: Creating a Local User for an Application 249](#_TOC_250050)

[Practice 19-3: Exploring OS and Password File Authentication 252](#_TOC_250049)

[Practices for Lesson 20: Configuring Privilege and Role Authorization 257](#_TOC_250048)

[Practices for Lesson 20: Overview 258](#_TOC_250047)

[Practice 20-1: Granting a Local Role (DBA) to PDBADMIN 259](#_TOC_250046)

[Practice 20-2: Using SQL\*Developer to Create Local Roles 262](#_TOC_250045)

[Practices for Lesson 21: Configuring User Resource Limits 271](#_TOC_250044)

[Practices for Lesson 21: Overview 272](#_TOC_250043)

[Practice 21-1: Using SQL\*Developer to Create a Local Profile 273](#_TOC_250042)

[Practice 21-2: Using SQL\*Developer to Create Local Users 284](#_TOC_250041)

[Practice 21-3: Configuring a Default Role for a User 296](#_TOC_250040)

[Practices for Lesson 22: Implementing Oracle Database Auditing 301](#_TOC_250039)

[Practices for Lesson 22: Overview 302](#_TOC_250038)

[Practice 22-1: Enabling Unified Auditing 303](#_TOC_250037)

[Practice 22-2: Creating Audit Users 307](#_TOC_250036)

[Practice 22-3: Creating an Audit Policy 309](#_TOC_250035)

[Practices for Lesson 23: Introduction to Loading and Transporting Data 313](#_TOC_250034)

[Practices for Lesson 23 314](#_TOC_250033)

[Practices for Lesson 24: Loading Data 315](#_TOC_250032)

[Practices for Lesson 24: Overview 316](#_TOC_250031)

[Practice 24-1: Loading Data into a PDB from an External File 317](#_TOC_250030)

[Practices for Lesson 25: Transporting Data 335](#_TOC_250029)

[Practices for Lesson 25: Overview 336](#_TOC_250028)

[Practice 25-1: Moving Data from One PDB to Another PDB 337](#_TOC_250027)

[Practice 25-2: Transporting a Tablespace 352](#_TOC_250026)

[Practices for Lesson 26: Using External Tables to Load and Transport Data 359](#_TOC_250025)

[Practices for Lesson 26: Overview 360](#_TOC_250024)

[Practice 26-1: Querying External Tables 361](#_TOC_250023)

[Practice 26-2: Unloading External Tables 369](#_TOC_250022)

[Practices for Lesson 27: Automated Maintenance Tasks Overview 373](#_TOC_250021)

[Practices for Lesson 27 374](#_TOC_250020)

[Practices for Lesson 28: Managing Tasks and Windows 375](#_TOC_250019)

[Practices for Lesson 28: Overview 376](#_TOC_250018)

[Practice 28-1: Enabling and Disabling Automated Maintenance Tasks 377](#_TOC_250017)

[Practice 28-2: Modifying the Duration of a Maintenance Window 379](#_TOC_250016)

[Practices for Lesson 29: Database Monitoring and Performance Tuning Overview 383](#_TOC_250015)

[Practices for Lesson 29 384](#_TOC_250014)

[Practices for Lesson 30: Monitoring Database Performance 385](#_TOC_250013)

[Practices for Lesson 30: Overview 386](#_TOC_250012)

[Practices for Lesson 31: Processes 399](#_TOC_250010)

[Practices for Lesson 31: Overview 400](#_TOC_250009)

[Practice 31-1: Examining the Database Background Processes 401](#_TOC_250008)

[Practice 31-2: Identifying the Database Server Processes 407](#_TOC_250007)

[Practices for Lesson 32: Tuning Database Memory 413](#_TOC_250006)

[Practices for Lesson 32: Overview 414](#_TOC_250005)

[Practice 32-1: Viewing Memory Configurations 415](#_TOC_250004)

[Practices for Lesson 33: Analyzing SQL and Optimizing Access Paths 421](#_TOC_250003)

[Practices for Lesson 33: Overview 422](#_TOC_250002)

[Practice 33-2: Using the Optimizer Statistics Advisor 440](#_TOC_250000)

# Course Practice Environment: Security Credentials

## 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 / orclpbd1 | LU | fenago |
| Database / orclpbd1 | 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 |

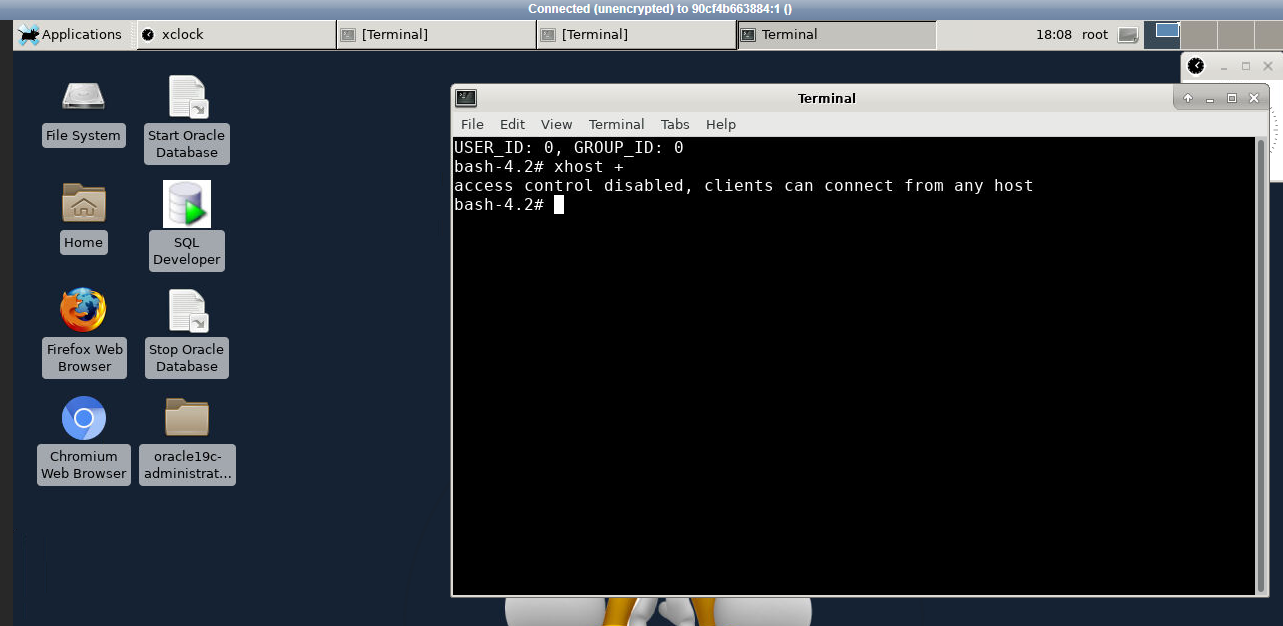
**Switch to oracle user from terminal**

### Overview

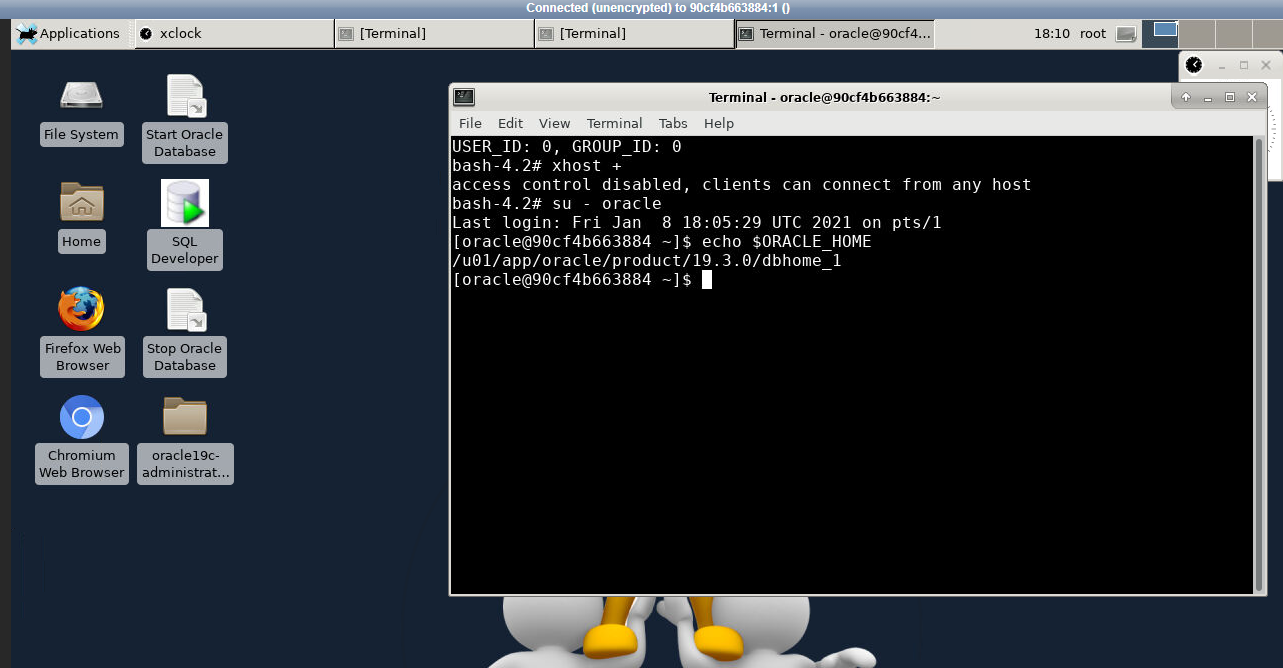
In this practice, you will switch to oracle user from terminal

### Tasks

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



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



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.
   1. Check the status of the database instance.

$ **pgrep -lf orclcdb**

$

* 1. If nothing is returned, start orclcdb database instance and listener. The listener is required for later tasks.

$ **dbstart.sh**

…

$

1. 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

$

1. 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

$

1. Create the CDB by using DBCA in silent mode.
   1. Change directory to /home/oracle/labs/DBMod\_CreateDB

**d /home/oracle/labs/DBMod\_CreateDB**

* 1. 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 ajoined with the word on the next line, e.g. -templateName or -useLocalUndoForPDBs

$ **gedit CrCDBTEST.sh**

…

$ORACLE\_HOME/bin/dbca -silent

templateName General\_Purpose.dbc

-createDatabase -

-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

* 1. Verify the edits

$ **cat CrCDBTEST.sh**

$ORACLE\_HOME/bin/dbca -silent

templateName General\_Purpose.dbc

-createDatabase -

-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

$

* 1. Change the permissions on the script to make it executable.

$ **chmod 755 CrCDBTEST.sh**

$

* 1. 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.

$

1. 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

$

1. Verify that the characteristics of the database are correct.
   1. 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>

* 1. 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>

* 1. 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 SYSAUX UNDOTBS1 TEMP

USERS

PERMANENT PERMANENT UNDO TEMPORARY

PERMANENT

SQL>

* 1. Verify that the port for EM Express is set correctly.

SQL> **SELECT dbms\_xdb\_config.gethttpsport() FROM dual;**

DBMS\_XDB\_CONFIG.GETHTTPSPORT()

5502

SQL>

1. 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**

$

1. Create the CDB by using the CREATE DATABASE command.
   1. Change the directory to /home/oracle/labs/DBMod\_CreateDB

$ **cd /home/oracle/labs/DBMod\_CreateDB**

$

* 1. 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;

* 1. 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

$

* 1. 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**

$

* 1. 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**

* 1. 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.

* 1. Save the modified file.
  2. 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

$

* 1. Start the database instance in NOMOUNT mode.

$ **sqlplus / AS SYSDBA**

Connected to an idle instance.

SQL> **STARTUP NOMOUNT**

... SQL>

* 1. 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>

* 1. 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.
  2. 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>

* 1. Exit from SQL\*Plus.

SQL> **EXIT**

**…**

$

1. 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**

$

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

$ **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 CDBDEV:/u01/app/oracle/product/19.3.0/dbhome\_1:N

$

1. Verify that the characteristics of the database are correct.
   1. Set the environment variables for your new database.

$ **. oraenv**

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

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

$

* 1. 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>

* 1. 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

ystem\_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.

* 1. 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 SYSAUX UNDOTBS1 TEMP

USERS

PERMANENT PERMANENT UNDO TEMPORARY

PERMANENT

SQL>

* 1. 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.

* 1. Exit SQL\*Plus.

SQ> **exit**

…

$

* 1. 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

$

1. 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

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

7 /

PL/SQL procedure successfully completed.

SQL> EXIT

…

$

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

$ **sqlplus / as sysdba**

… SQL>

1. Shut down the database instance in 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>

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

SQL> **SHOW USER** USER is "SYS" SQL>

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

SQL> **show con\_name**

ERROR:

ORA-01034: ORACLE not available Process ID: 0

Session ID: 0 Serial number: 0

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

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

SQL> **startup nomount;**

ORACLE instance started.

Total System Global Area 2013264224 bytes Fixed Size 9136480 bytes Variable Size 637534208 bytes

Database Buffers 1358954496 bytes Redo Buffers 7639040 bytes

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

SQL> alter database mount;

Database altered.

SQL>

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

SQL> **alter database open;**

Database altered.

SQL>

1. Show the current container name.

SQL> **show con\_name**

CON\_NAME

CDB$ROOT SQL>

1. Show the current user.

SQL> **show user**

USER is "SYS"

SQL>

1. Check whether 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**  CON\_ID NAME OPEN\_MODE | | | **FROM** | **v$pdbs;** |
| 2 | PDB$SEED | READ ONLY | | |
| 3 | ORCLPDB1 | READ WRITE | | |
| 4 | ORCLPDB2 | READ WRITE | | |
| 5 | ORCLPDB3 | MOUNTED | | |
| SQL> |  |  | | |

1. 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

1. ORCLPDB1
2. ORCLPDB2

OPEN

OPEN

SQL>

1. Remove the saved states.
   1. Set the pluggable databases closed.

SQL> **alter pluggable database all close;**

Pluggable database altered. SQL>

* 1. Set the saved states

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

Pluggable database altered. SQL>

* 1. View the PDB states.

SQL> **SELECT con\_id, con\_name, state FROM DBA\_PDB\_SAVED\_STATES;**

no rows selected SQL>

1. Verify the behavior of the save states:
   1. Shut down the CDB.

SQL> **SHUTDOWN IMMEDIATE**

Database closed. Database dismounted.

ORACLE instance shut down.

**…**

SQL>

* 1. Start up the CDB.

SQL> **STARTUP**

**… SQL>**

* 1. 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> |  |  |  |  |

1. Reset the PDB save states to open:
   1. Set the pluggable databases open.

SQL> **alter pluggable database all open;**

Pluggable database altered. SQL>

* 1. Set the saved state for all PDBs.

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

Pluggable database altered. SQL>

* 1. 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> |  |  |  |  |  |

1. Shut down a single PDB, ORCLPDB3.

SQL> **alter pluggable database orclpdb3 close;**

Pluggable database altered. SQL>

1. View the status of the PDBs.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 3 | ORCLPDB1 | OPEN |
| 4 | ORCLPDB2 | OPEN |
| SQL> | 5 | ORCLPDB3 | OPEN |

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

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

Pluggable database dropped. SQL>

SQL> show pdbs

CON\_ID CON\_NAME

STATE

1. Exit SQL\*Plus.

SAL> **exit**

…

$

1. 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

$

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

privilege.

$ **sqlplus / as sysdba**

… SQL>

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

SQL> **show parameter spfile**

NAME

TYPE

VALUE

spfile

string

/u01/app/oracle/product/19.3.0/dbhome\_1/dbs/spfileorclcdb.ora

SQL>

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

SQL> **host**

$

* 1. 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 hc\_orclcdb.dat hc\_rcatcdb.dat initCDBDEV.ora | lkCDBDEV lkCDBTEST lkORCLCDB lkRCATCDB | orapworclcdb orapwrcatcdb snapcf\_orclcdb.f spfileCDBTEST.ora | spfilercatcdb.ora |
| $ |  |  |  |

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

* 1. Use the cat or more command to view the contents of the sample text initialization parameter file (PFILE), init.ora. 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. |
| # hpiao | | 06/05/97 | - fix for 803 |
| # glavash | | 05/12/97 | - add oracle\_trace\_enable comment |
| # hpiao | | 04/22/97 | - remove ifile=, events=, etc. |
| # alingelb | | 09/19/94 | - remove vms-specific stuff |
| # dpawson start | | 07/07/93 | - add more comments regarded archive |
| # 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**

* 1. 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**

$

* 1. 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'

$

* 1. Return to SQL\*Plus.

$ **exit**

SQL>

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

SQL> **shutdown immediate**

Database closed. Database dismounted.

ORACLE instance shut down.

SQL>

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

SQL> **host**

$

* 1. Change to the $ORACLE\_HOME/dbs directory.

$ **cd $ORACLE\_HOME/dbs**

$

* 1. 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**

$

* 1. Return to SQL\*Plus.

$ **exit**

SQL>

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

SQL> **startup**

…

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

SQL> **show parameter spfile**

NAME

TYPE

VALUE

spfile

string

SQL>

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

SQL> **shutdown immediate**

Database closed. Database dismounted.

ORACLE instance shut down.

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

SQL> **host**

$

* 1. Change to the $ORACLE\_HOME/dbs directory.

$ **cd $ORACLE\_HOME/dbs**

$

* 1. Rename the orig\_spfileorclcdb.ora file to spfileorclcdb.ora.

$ **mv orig\_spfileorclcdb.ora spfileorclcdb.ora**

$

* 1. Return to SQL\*Plus.

$ **exit**

SQL>

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

SQL> **startup**

…

* 1. 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>

1. 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

$

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

privilege.

$ **sqlplus / as sysdba**

… SQL>

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SQL> **show**  NAME | **parameter** | **db\_name** |  | TYPE |  | VALUE |
| db\_name  SQL> |  |  |  | string |  | orclcdb |

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

SQL> **show parameter db\_domain**

NAME

TYPE

VALUE

db\_domain

string

SQL>

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

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

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

#### SQL> show parameter db\_recovery\_file\_dest

|  |  |  |  |
| --- | --- | --- | --- |
| NAME |  | TYPE | VALUE |
| --  db\_recovery\_file\_dest |  | string | /u01/app/oracle/fast\_recove |
|  |  |  | ry\_area |

db\_recovery\_file\_dest\_size big integer 14970M

SQL>

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

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

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

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

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

The range of values for SGA\_TARGET can be from 64 MB to an operating system- dependent value.

SQL> **show parameter sga**

NAME

TYPE

VALUE

allow\_group\_access\_to\_sga lock\_sga

pre\_page\_sga sga\_max\_size sga\_min\_size sga\_target

unified\_audit\_sga\_queue\_size

boolean

boolean boolean

FALSE

FALSE TRUE

big integer 1920M

big integer 0 big integer 1920M

integer

1048576

SQL>

1. View the UNDO\_TABLESPACE parameter. This parameter specifies the undo tablespace to be used when an instance starts. Oracle Database creates and manages information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo and are stored in the undo tablespace. The results below indicate that the undo tablespace in your environment is UNDOTBS1.
2. 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

SQL> **show parameter undo\_tablespace**

NAME

TYPE

VALUE

undo\_tablespace

string

UNDOTBS1

SQL>

|  |  |  |
| --- | --- | --- |
| **compatible** | **string** | **19.0.0** |
| noncdb\_compatible | boolean | FALSE |

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

SQL> **show parameter control\_files**

NAME TYPE

VALUE

control\_files

string

/u01/app/oracle/oradat a/ORCLCDB/control01.ct l, /u01/app/oracle/fas t\_recovery\_area/ORCLCD

B/control02.ctl

SQL>

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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> |  |  |  |  |

* 1. View the SESSIONS parameter. This parameter specifies the maximum number of sessions that can be created in the system. Because every login requires a session,

this parameter effectively determines the maximum number of concurrent users in the system. Notice in the results that the session entry has a value of 472. You should always set this parameter explicitly to a value equivalent to your estimate of the maximum number of concurrent users, plus the number of background processes, plus approximately 10% for recursive sessions.

SQL> **show parameter sessions**

NAME TYPE

VALUE

java\_max\_sessionspace\_size

integer

java\_soft\_sessionspace\_limit integer

license\_max\_sessions license\_sessions\_warning **sessions**

shared\_server\_sessions

integer integer **integer**

integer

0

0

0

0

**472**

SQL>

#### View Advanced Parameters

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

1. 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

transactions\_per\_rollback\_segment integer

519

5

SQL>

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

SQL> **show parameter db\_files**

NAME

TYPE

VALUE

db\_files

integer

200

SQL>

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

SQL> **show parameter commit\_logging**

NAME

TYPE

VALUE

commit\_logging

string

SQL>

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

SQL> **show parameter commit\_wait**

NAME

TYPE

VALUE

commit\_wait

string

SQL>

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

SQL> **show parameter shared\_pool\_size**

NAME

TYPE

VALUE

shared\_pool\_size

big integer 0

SQL>

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

SQL> **show parameter db\_block\_size**

NAME

TYPE

VALUE

db\_block\_size

integer

8192

SQL>

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

SQL> **show parameter db\_cache\_size**

NAME

TYPE

VALUE

db\_cache\_size

big integer 0

SQL>

1. View the UNDO\_MANAGEMENT parameter. This parameter specifies the undo space management mode that the system should use. When set to AUTO, the instance is started in automatic undo management mode. Otherwise, it is started in rollback undo mode. In rollback undo mode, undo space is allocated as rollback segments. In automatic undo mode, undo space is allocated as undo tablespaces. The value is AUTO or MANUAL. If the UNDO\_MANAGEMENT parameter is omitted when the instance is started, the default value AUTO is used.
2. 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.

SQL> **show parameter undo\_management**

NAME

TYPE

VALUE

**undo\_management**

string

AUTO

SQL>

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

* 1. View the MEMORY\_TARGET parameter.

SQL> **show parameter memory\_target**

NAME

TYPE

VALUE

memory\_target

big integer 0

SQL>

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

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

SQL> **show parameter memory\_max\_target**

NAME

TYPE

VALUE

memory\_max\_target

big integer 0

SQL>

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.

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

SQL> **set 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>

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

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SQL> **describe**  Name | **v$parameter** |  | 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 ISADJUSTED ISDEPRECATED ISBASIC DESCRIPTION UPDATE\_COMMENT HASH

CON\_ID

VARCHAR2(10) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(255) VARCHAR2(255) NUMBER

NUMBER

SQL>

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

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

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

SQL> **col name format a35**

SQL> **col value format a20**

SQL> **select name, issys\_modifiable, value from v$parameter**

**order by name;**

NAME

ISSYS\_MOD VALUE

DBFIPS\_140

active\_instance\_count

… transactions

transactions\_per\_rollback\_segment

… wallet\_root

workarea\_size\_policy

xml\_db\_events

FALSE

FALSE

FALSE

FALSE FALSE

519

5

FALSE IMMEDIATE AUTO

IMMEDIATE enable

445 rows selected.

SQL>

* 1. Query the V$PARAMETER view again, but this time be more specific. Include a WHERE clause to specify all parameters that contain the word "pool." The query returns 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 large\_pool\_size java\_pool\_size streams\_pool\_size shared\_pool\_reserved\_size memoptimize\_pool\_size buffer\_pool\_keep buffer\_pool\_recycle

olap\_page\_pool\_size

0

0

0

0

30198988

0

0

9 rows selected.

SQL>

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

SQL> **describe v$spparameter**

Name Null? Type

FAMILY SID NAME TYPE VALUE

DISPLAY\_VALUE ISSPECIFIED ORDINALV

UPDATE\_COMMENT

VARCHAR2(80) VARCHAR2(80) VARCHAR2(80) VARCHAR2(11) VARCHAR2(255) VARCHAR2(255) VARCHAR2(6) NUMBER

VARCHAR2(255)

CON\_ID

NUMBER

SQL>

* 1. 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>

1. 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.
   1. 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 CON\_ID

VARCHAR2(255) NUMBER

SQL>

* 1. 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>

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

command.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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>

* 1. 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>

1. 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 orclcdb 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**

…

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

SQL> **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

ISSES ISSYS\_MOD ISPDB VALUE

nls\_date\_format

TRUE FALSE

TRUE

1. 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

1. Connect to ORCLPDB1. Run a simple query against the sample data to view an example of the current default date format in use.
   1. Switch to ORCLPDB1 by using the ALTER SESSION command.

SQL> **ALTER SESSION SET container = ORCLPDB1;**

Session altered.

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

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

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

LAST\_NAME

HIRE\_DATE

King Kochhar

… Higgins

Gietz

17-JUN-03

21-SEP-05

07-JUN-02

07-JUN-02

107 rows selected.

ALTER SESSION command.

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

Session altered.

1. 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;**

1. using the SHOW PARAMETER command.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LAST\_NAME HIRE\_DATE | | | | | |
| King |  |  | jun | 17 | 2003 |
| Kochhar  … Higgins |  |  | sep  jun | 21  07 | 2005  2002 |
| Gietz |  |  | jun | 07 | 2002 |
| 107 rows | selected. |  |  |  |  |
| Query the NLS\_DATE\_FORMAT parameter again by The value column now reflects the custom date form | | | | | |

at.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SQL> **SHOW PARAMETER**  NAME | **nls\_date\_format** |  | TYPE |  | VALUE |
| nls\_date\_format |  |  | string |  | mon dd yyyy |

1. Disconnect from ORCLPDB1 to end your session.

SQL> **DISCONNECT**

**...**

1. 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>.
   1. Get the full hostname.

SQL> **! hostname -f**

edvmr1p0.us.oracle.com

* 1. 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 tcp| 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))

* 1. Connect as a user with sysdba privileges.

SQL> **Connect / as sysdba**

**…**

* 1. 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

* 1. 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>

1. 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 |
| … Higgins |  |  | 07-JUN-02 |
| Gietz |  |  | 07-JUN-02 |
| 107 rows | selected. |  |  |

1. 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**

…

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

SQL> **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

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

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

System altered.

1. Use the SHOW PARAMETER command to verify that the JOB\_QUEUE\_PROCESSES parameter value is now equal to 15. Notice that only job was entered with the SHOW PARAMETER command instead of the full name, job\_queue\_processes. Remember, when you use the SHOW PARAMETER command, you don't have to enter the full name. The database server will find all parameters that contain the letters job. In this example, the database server found 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 max\_datapump\_parallel\_per\_job | integer string | 50 | 100 |

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

SQL> **SHUTDOWN IMMEDIATE**

Database closed. Database dismounted.

ORACLE instance shut down.

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

SQL> **STARTUP**

…

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

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 |

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

* 1. 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.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL> **SHOW PARAMETER sec\_max**  NAME |  | TYPE |  | VALUE |
| sec\_max\_failed\_login\_attempts |  | integer |  | 3 |

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

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

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

SQL> **shutdown immediate**

Database closed. Database dismounted.

ORACLE instance shut down.

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

SQL> **startup**

…

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL> **SHOW PARAMETER sec\_max**  NAME |  | TYPE |  | VALUE |
| sec\_max\_failed\_login\_attempts |  | integer |  | 2 |

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

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.

1. 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.

1. 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**

…

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

Diag Incident

Diag Cdump Health Monitor

TRUE

/u01/app/oracle

/u01/app/oracle/diag/rdbms/orclcdb/orclcdb

/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace

/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert

/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/incident

/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/cdump

/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.

1. 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.
   1. Browse to the /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert

directory.

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

* 1. 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

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

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

$ **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>

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

directory.

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

* 1. 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

* 1. 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 SCHEDULER[0x4D52]:DEFAULT\_MAINTENANCE\_PLAN via scheduler window

Setting Resource Manager CDB plan DEFAULT\_MAINTENANCE\_PLAN via parameter

* 1. 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>

1. 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/edvmr1p0/listener

6: diag/clients/user\_oracle/host\_3132364359\_110 Q: to quit

Please select option: **2**

1. 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

1. 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

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

…

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

…

ALTER DATABASE MOUNT

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

…

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

…

Completed: ALTER DATABASE MOUNT

ALTER DATABASE OPEN

…

1. Exit the vi editor by entering **:q!** and pressing **Enter**.
2. At Alert log list, enter **Q** to leave alert log list.
3. 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.
   1. Start SQL\*Plus and log in to the database as the SYS user with the SYSDBA privilege.

$ **sqlplus / as sysdba**

…

* 1. Switch to PDB1.

SQL> **ALTER SESSION SET CONTAINER = ORCLPDB1;**

Session altered.

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

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

NAME

TYPE

VALUE

enable\_ddl\_logging

boolean

FALSE

* 1. 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>

1. 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.

1. Exit SQL\*Plus.

SQL> **EXIT**

…

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

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

1. List the contents of the log directory.

$ **ls**

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

1. 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

1. 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>

…

1. 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

$

1. 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

$

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

$ORACLE\_HOME/network/admin directory.

* 1. 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

$

* 1. Copy the tnsnames.ora file to tnsnames.old.

$ **cp tnsnames.ora tnsnames.old**

$

* 1. 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 |
| $ |  |  |  |  |  |  |  |  |

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

$ **hostname -f**

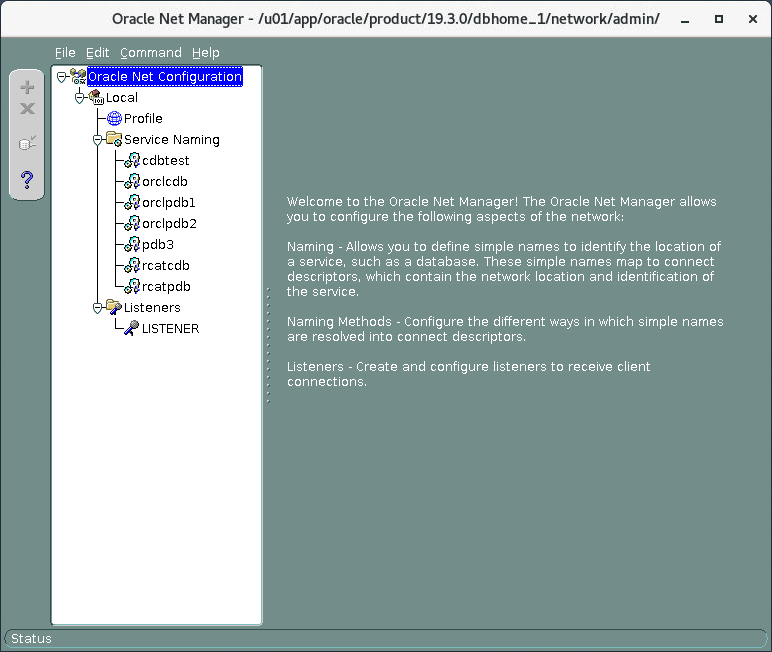
edvmr1p0.us.oracle.com

$

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

* 1. Invoke Oracle Net Manager.

$ **netmgr**



* 1. Expand **Local** and select **Service Naming**.
  2. Click the green **plus sign**.
  3. In the Service Name field, enter **testorcl** and then click **Next**.
  4. Select **TCP/IP** and then click **Next**.
  5. In the Host Name field, enter the fully qualified **host name** you found in Step 4.
  6. In the Port Number field, enter **1521** and then click **Next**.
  7. In the Service field, enter **CDBTEST** .
  8. Under Connection type, select Dedicated Server and then click **Next**.
  9. Click **Test**.
  10. In the “Connection test” dialog box, the test will fail because SCOTT is not in the default database. Click **Change Login.**
  11. In Change Login Box, enter username **system** and **password.** See *Appendix - Product-Specific Credentials* for the password. Click **OK.**
  12. Click **Test**.
  13. When "The connection test was successful" message appears, click **Close** and then

#### Finish.

* 1. Click **File > Save Network Configuration**.
  2. Exit Oracle Net Manager.

1. 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.
   1. Ensure your environment is set for the orclcdb database by executing the oraenv

command.

* 1. Invoke SQL\*Plus and connect by using the testorcl service name.

$ **sqlplus system@testorcl**

Enter password: ***password***

SQL>

1. 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

edvmr1p0

SQL>

1. 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

$

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

$ **cd $ORACLE\_HOME/network/admin**

$

* 1. 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 |

* 1. 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 = edvmr1p0)(PORT = 1521))

)

(CONNECT\_DATA = (SERVER = DEDICATED)

(SERVICE\_NAME = CDBTEST)

)

)

LISTENER\_CDBTEST =

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

$

1. Create a net service name, MyPDB1, for ORCLPDB1 by using Oracle Net Manager.
   1. Invoke Oracle Net Manager.

$ **netmgr**

* 1. Expand **Local** and select **Service Naming**.
  2. Click the green **plus sign**.
  3. In the Service Name field, enter **MY1PDB1** and then click **Next**.
  4. Select **TCP/IP** and then click **Next**.
  5. 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**.
  6. In the Service field, enter **ORCLPDB1** .
  7. Under Connection type, select Dedicated Server and then click **Next**.
  8. Click **Test**.
  9. 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.**
  10. Click **Test**.
  11. When " The connection test was successful" message appears, click **Close** and then

#### Finish.

* 1. Click **File > Save Network Configuration**.
  2. Exit Oracle Net Manager.

1. Verify that the entry has been added to the tnsnames.ora file.
   1. Change the directory to $ORACLE\_HOME/network/admin.

$ **cd $ORACLE\_HOME/network/admin**

$

* 1. 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)

)

…

$

1. 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)

$

1. Connect to ORCLPDB1 and verify the container.
   1. 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>

* 1. Verify that the current container name is ORCLPDB1.

SQL> **SHOW con\_name**

CON\_NAME

ORCLPDB1 SQL>

1. Exit SQL\*Plus.

SQL> exit

…

$

1. 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

$

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

$ **sqlplus / as sysdba**

… SQL>

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SQL> **SHOW PARAMETER**  NAME | **INSTANCE\_NAME** |  | TYPE |  | VALUE |
| instance\_name  SQL> |  |  | string |  | orclcdb |

* 1. SERVICE\_NAMES: This parameter identifies the service names that users can use in their connection strings to connect to the database instance. By default, the service name takes on the same name as the global database name, 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

service\_names

string

orclcdb

SQL>

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

NAME

TYPE

VALUE

local\_listener

string

LISTENER\_ORCLCDB

SQL>

SQL> **show parameter remote\_listener**

NAME

TYPE

VALUE

remote\_listener

string

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. View the server-side tnsnames.ora file and locate the entry that resolves the

LOCAL\_LISTENER parameter value, which is the LISTENER\_ORCLCDB alias.

* 1. Change directories to $ORACLE\_HOME/network/admin.

$ **cd $ORACLE\_HOME/network/admin**

$

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

$ **ls**

listener.ora samples shrept.lst tnsnames.old tnsnames.ora

$

* 1. 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.
  2. **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.o ra

# 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 = edvmr1p0)(PORT = 1521))

$

1. 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 = edvmr1p0)(PORT = 1521))

)

ADR\_BASE\_LISTENER = /u01/app/oracle

$

1. 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>

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

LSNRCTL> **help**

The following operations are available

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

start servacls trace

set\*

stop version spawn

show\*

status reload

quit

services save\_config

exit

LSNRCTL>

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

LSNRCTL> **show current\_listener**

Current Listener is LISTENER

LSNRCL>

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

points, the wallet directory, and a list of registered services and whether they are ready.

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.o ra

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.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))

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>

* 1. **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.

* 1. 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: edvmr1p0, pid: 27937>

(ADDRESS=(PROTOCOL=tcp)(HOST=edvmr1p0.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: edvmr1p0, pid: 30421>

(ADDRESS=(PROTOCOL=tcp)(HOST=edvmr1p0.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: edvmr1p0, pid: 24346>

(ADDRESS=(PROTOCOL=tcp)(HOST=edvmr1p0.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: edvmr1p0, 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.

* 1. 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>

1. 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.
   1. Set your environment variables using oraenv to orclcdb

$ **. oraenv**

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

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

$

* 1. Obtain your host name and domain. The format is host.domain

$ **hostname -f**

edvmr1p0.us.oracle.com

$

* 1. Browse to $ORACLE\_HOME/network/admin.

$ **cd $ORACLE\_HOME/network/admin**

$

* 1. Copy the tnsnames.ora file to tnsnames.ora.3-2 and then open tnsnames.ora

in gedit.

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

$

* 1. 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.o ra

# 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))

...

* 1. Save the file and then exit the editor. Hint: vi wite & quit is **:wq!**

1. Modify the LOCAL\_LISTENER initialization parameter to include both LISTENER\_ORCLCDB

and LISTENER2 aliases.

* 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

$

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

$ **sqlplus / as sysdba**

… SQL>

* 1. 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>

* 1. 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>

* 1. 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>

* 1. 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>

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

SQL> **exit**

…

$

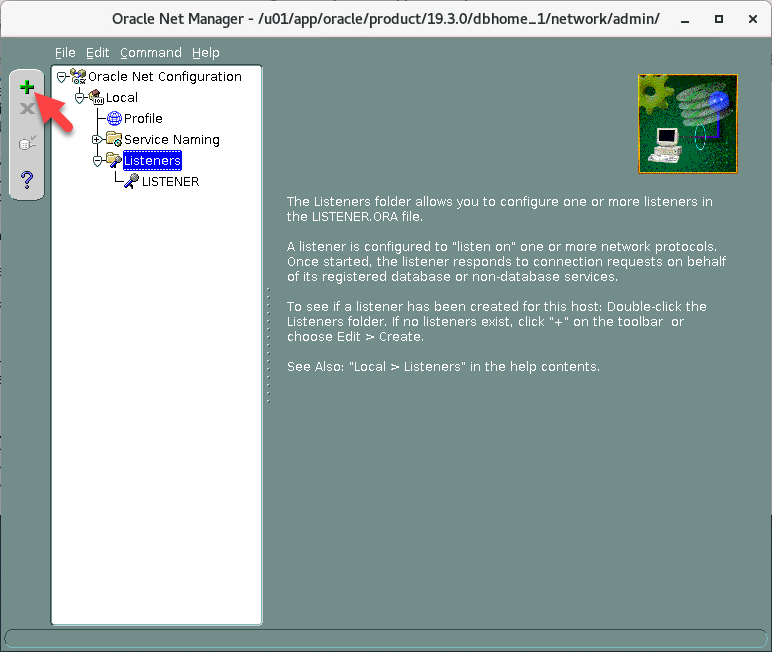
1. To manage LISTENER2 with the Listener Control utility, you need to add an entry in the listener.ora file so that the utility knows how to connect to LISTENER2. In this task, you will use Oracle Net Manager to add the entry. It's important to remember that dynamic service registration does not make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

$ **cd $ORACLE\_HOME/network/admin**

$ **cp listener.ora listener.old**

$

* 1. In the terminal window, start Oracle Net Manager. Use **netmgr** command.
  2. 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.



* 1. In the Choose Listener Name dialog box, enter **LISTENER2** and click **OK**. LISTENER2

is added to the list of listeners.

* 1. With LISTENER2 selected on the left side, click **Add Address**.
  2. 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**.
  3. For interest, select **General Parameters** from the drop-down list. Review the configuration options on the General, Logging & Tracing, and Authentication tabs.
  4. Select **Database Services** from the drop-down list. There are no databases currently configured for the listener.
  5. Select **File** and then **Save Network Configuration**.
  6. Select **File** and then **Exit** to exit Oracle Net Manager. You just added an entry into the

listeners.ora file.

* 1. Change directory (cd) to **$ORACLE\_HOME/network/admin**

(/u01/app/oracle/product/<version>/dbhome\_1/network/admin).

* 1. 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.o ra

# 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

$

1. Using the Listener Control utility, check the status of LISTENER2.
   1. In the terminal window, start the Listener Control utility.

$ **lsnrctl**

… LSNRCTL>

* 1. 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>

* 1. 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.o ra

Listener Log File

/u01/app/oracle/diag/tnslsnr/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>

* 1. 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.

* 1. 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.o ra

Listener Log File

/u01/app/oracle/diag/tnslsnr/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>

* 1. 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>

1. 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

$

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

$ **sqlplus / as sysdba**

**…**

SQL>

1. Determine whether the shared server architecture is implemented in your database.
   1. Check the value of the SHARED\_SERVER initialization parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL> **show parameter shared\_server** | | | | |
| NAME |  | TYPE |  | VALUE |
| max\_shared\_servers shared\_server\_sessions shared\_servers |  | integer integer integer |  | 1 |
| SQL> |  |  |  |  |

* 1. Check the value of the DISPATCHERS initialization parameter.

SQL> **show parameter dispatchers**

NAME

TYPE

VALUE

dispatchers

max\_dispatchers

string (PROTOCOL=TCP) (SERVICE=orclcdbXDB)

integer

SQL>

* 1. 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.

1. Enable three shared servers in your database.

SQL> **alter system set shared\_servers = 3;**

System altered.

SQL>

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

SQL> **show parameter dispatchers**

NAME

TYPE

VALUE

dispatchers

max\_dispatchers

string (PROTOCOL=TCP) (SERVICE=orclcdbXDB)

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.

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

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

System altered.

1. Confirm change:

SQL> **show parameter dispatchers**

NAME

TYPE

VALUE

dispatchers

max\_dispatchers

string (PROTOCOL=TCP)

integer

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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

$

1. Configure a network service that uses a shared server connection.
   1. 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

$

* 1. Make a copy of tnsnames.ora.

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

$

* 1. 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)**

**)**

**)**

1. 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>

1. 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

$

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

$ **sqlplus / as sysdba**

… SQL>

1. 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

000000009F967A18 NONE

SQL>

1. In **Terminal 1**, log out of the SQL\*Plus session that uses the test\_ss service.
2. 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>

1. 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

$

1. 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

$

1. As the oracle OS user, create the new PDB.
   1. Log in to SQL\*Plus and connect to the CDB root with a user with the CREATE PLUGGABLE DATABASE privilege.

$ **sqlplus / as sysdba**

… SQL>

* 1. 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>

1. 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

2 PDB$SEED

1. ORCLPDB2
2. ORCLPDB3

NORMAL NORMAL NORMAL

NEW

SQL>

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

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

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

SQL> **alter pluggable database orclpdb3 open;**

Pluggable database altered.

SQL> **select pdb\_id, pdb\_name, status from cdb\_pdbs;**

PDB\_ID PDB\_NAME

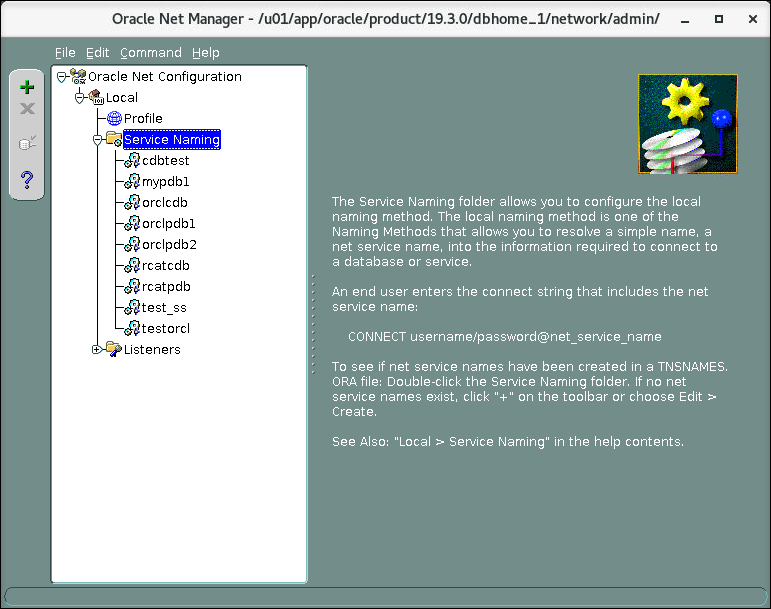
STATUS

SQL> **exit**

* 1. Launch netmgr.

$ **netmgr**

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



* 1. Click the **plus sign**.
  2. Enter Net Service Name: **PDB3**, and click **Next**.
  3. Select **TCP/IP (Internet Protocol)**, and click **Next**.
  4. Enter Host Name: **localhost**, verify the Port Number is **1521**, and click **Next**.
  5. Enter Service Name: **ORCLPDB3**
  6. For Connection Type, select **Dedicated Server** and click **Next**.
  7. Click **Finish**.

#### Click File>Save Network Configuration.

* 1. Click **File>Exit**.

1. 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>

1. 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**

…

$

1. 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**

…

$

1. 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

$

1. 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

…

$

1. 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**

…

$

1. 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>

1. In CDBTEST, clone PDB\_HOTCLONE from PDB\_SOURCE\_FOR\_HOTCLONE in hot mode.
   1. 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**

$

* 1. 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>

* 1. 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>

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

CDBTEST> **ALTER PLUGGABLE DATABASE pdb\_hotclone OPEN READ ONLY;**

Pluggable database altered. CDBTEST>

1. 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>

1. 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>

1. 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>

1. 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.

1. 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**

…

$

1. In *Session CDBTEST*, check that the data in PDB\_HOTCLONE is refreshed.
   1. 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>

* 1. Close PDB\_HOTCLONE.

CDBTEST> **ALTER PLUGGABLE DATABASE pdb\_hotclone CLOSE;**

Pluggable database altered. CDBTEST>

* 1. 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**

…

$

1. 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**

…

$

1. 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***

…

$

1. 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>

1. 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>

1. In *Session ORCLCDB1*, prepare to relocate ORCLPDB3 from orclcdb into CDBTEST as

PDB\_RELOCATED.

* 1. 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>

* 1. 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

1. PDB$SEED
2. ORCLPDB1
3. ORCLPDB2

6 ORCLPDB3

READ ONLY NO

READ WRITE NO READ WRITE NO

READ WRITE NO

ORCLCDB1>

* 1. 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>

* 1. Re-display the status of all the pluggable databases

ORCLCDB1> **show pdbs**

CON\_ID CON\_NAME

OPEN MODE RESTRICTED

1. PDB$SEED
2. ORCLPDB1
3. ORCLPDB2
4. ORCLPDB3

READ ONLY NO READ WRITE NO READ WRITE NO

READ WRITE NO

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

Pluggable database altered. ORCLCDB1>

1. Open a new terminal, and set the title to *Session CDBTEST.*
   1. 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>

* 1. 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>

* 1. 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>

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

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

Grant succeeded.

CDBTEST>

1. 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>

1. 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*.

1. 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

1. for i in 1..5000 loop
2. insert into test.bigtab values ('NEW DATA during relocation');
3. dbms\_lock.sleep(1);
4. commit;
5. end loop;
6. end;

8 /

…

1. During **sessions.sh** execution:
   1. 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 PDB$SEED ORCLPDB2

ORCLPDB3

NORMAL NORMAL NORMAL

NORMAL

ORCLCDB1>

* 1. 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

NEW DATA during relocation

10000

102

CDBTEST>

1. 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.
   1. The source PDB is closed and dropped from the source CDB.
   2. 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

…

$

1. 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

NEW DATA during relocation

10000

221

CDBTEST> **select pdb\_name, status from cdb\_pdbs;**

PDB\_NAME

STATUS

PDB\_RELOCATED

NORMAL

CDBTEST>

1. 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

PDB$SEED ORCLPDB2

NORMAL

NORMAL NORMAL

ORCLCDB1> **exit**

…

$

1. 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**

…

$

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

$ **sqlplus / as sysdba**

SQL> **revoke sysoper from system container = all;**

Revoke succeeded.

SQL> **exit**

**…**

**$**

1. 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.

1. 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> |  |  |  |  |  |  |  |

1. 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>

1. Close ORCLPDB3.

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

Pluggable database altered.

1. 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>

1. 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>

1. Open PDB3\_ORCL.

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

Pluggable database altered.

SQL> **alter pluggable database open;**

Pluggable database altered. SQL>

1. Verify that PDB3\_ORCL is in READ WRITE mode, and then exit sqlplus.
2. Add the PDB3\_ORCL service name to the tnsnames.ora file with the script

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**

**…**

$

$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

$

1. 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

1. 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>

1. 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  SQL> |  | boolean |  | TRUE |

1. 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>

1. 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>

1. Create another PDB and check the parameter value in this new PDB in the same CDB.
   1. 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>

* 1. 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>

* 1. 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>

1. 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>

1. Check the parameter value after CDB shutdown/startup in both the CDB root and PDBs.
   1. 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>

* 1. 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

1. PDB$SEED
2. ORCLPDB1
3. ORCLPDB2
4. ORCLPDB3
5. TEST

READ ONLY NO READ WRITE NO READ WRITE NO READ WRITE NO

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>

1. Drop PDBs, test and PDB3\_ORCL.

SQL> **show pdbs**

CON\_ID CON\_NAME

OPEN MODE RESTRICTED

2 PDB$SEED

6 PDB3\_ORCL

1. ORCLPDB2
2. TEST

3 ORCLPDB1

READ ONLY NO

READ WRITE NO READ WRITE NO READ WRITE NO

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>

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

SQL> **show pdbs**

SQL> **exit**

…

$

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CON\_ID |  | CON\_NAME |  | OPEN | MODE |  | RESTRICTED |
| 2 |  | PDB$SEED |  | READ | ONLY |  | NO |
| 3 |  | ORCLPDB1 |  | READ | WRITE |  | NO |
| 4 |  | ORCLPDB2 |  | READ | WRITE |  | NO |

1. 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

$

1. Grant DBA to PDBADMIN in ORCLPDB1
   1. 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>

* 1. 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;

1. 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>

1. List the columns in the DBA\_TABLESPACES view by using the DESCRIBE command.

SQL> **describe dba\_tablespaces**

Name

Null?

Type

TABLESPACE\_NAME

BLOCK\_SIZE

NOT NULL VARCHAR2(30)

NOT NULL NUMBER

INITIAL\_EXTENT NEXT\_EXTENT MIN\_EXTENTS

…

DEF\_CELLMEMORY DEF\_INMEMORY\_SERVICE DEF\_INMEMORY\_SERVICE\_NAME LOST\_WRITE\_PROTECT

CHUNK\_TABLESPACE

NUMBER NUMBER

NOT NULL NUMBER

VARCHAR2(14) VARCHAR2(12) VARCHAR2(1000)

VARCHAR2(7)

VARCHAR2(1)

SQL>

1. 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>

1. Find out which tablespace contains the HR schema by querying the ALL\_TABLES view.

SQL> **select distinct tablespace\_name from all\_tables where owner='HR';**

TABLESPACE\_NAME

USERS SQL>

1. Query the STATUS, CONTENTS, LOGGING, PLUGGED\_IN, BIGFILE, EXTENT\_MANAGEMENT, and ALLOCATION\_TYPE columns in the DBA\_TABLESPACES view for the SYSAUX tablespace.

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

ONLINE

PERMANENT

LOGGING NO NO LOCAL

SYSTEM

SQL>

* STATUS shows the value ONLINE, indicating the tablespace is available to users.
* CONTENTS indicates the PERMANENT tablespace type.
* LOGGING shows the value LOGGING, indicating that certain DML operations are logged in the redo log file.
* PLUGGED\_IN shows the value NO, indicating that the tablespace is not plugged in.
* BIGFILE shows the value NO, indicating that the tablespace is a smallfile tablespace.
* EXTENT\_MANAGEMENT shows the value LOCAL, indicating that the tablespace is locally managed (not dictionary managed).
* ALLOCATION\_TYPE shows the value SYSTEM, indicating that the extents of the tablespace are managed by the system, and you cannot specify an extent size.

1. 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 INCLUDED\_IN\_DATABASE\_BACKUP BIGFILE  FLASHBACK\_ON ENCRYPT\_IN\_BACKUP CON\_ID |  |  |  | VARCHAR2(30) VARCHAR2(3) VARCHAR2(3) VARCHAR2(3) VARCHAR2(3) NUMBER |
| SQL> | | | | |

1. 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.

1. 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>

1. 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>

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

SQL> **describe dba\_data\_files**

Name

Null?

Type

FILE\_NAME FILE\_ID TABLESPACE\_NAME BYTES

BLOCKS STATUS

RELATIVE\_FNO

VARCHAR2(513) NUMBER VARCHAR2(30) NUMBER NUMBER VARCHAR2(9)

NUMBER

AUTOEXTENSIBLE MAXBYTES MAXBLOCKS INCREMENT\_BY USER\_BYTES USER\_BLOCKS ONLINE\_STATUS

LOST\_WRITE\_PROTECT

VARCHAR2(3) NUMBER NUMBER NUMBER NUMBER NUMBER VARCHAR2(7)

VARCHAR2(7)

SQL>

1. 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.

1. 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

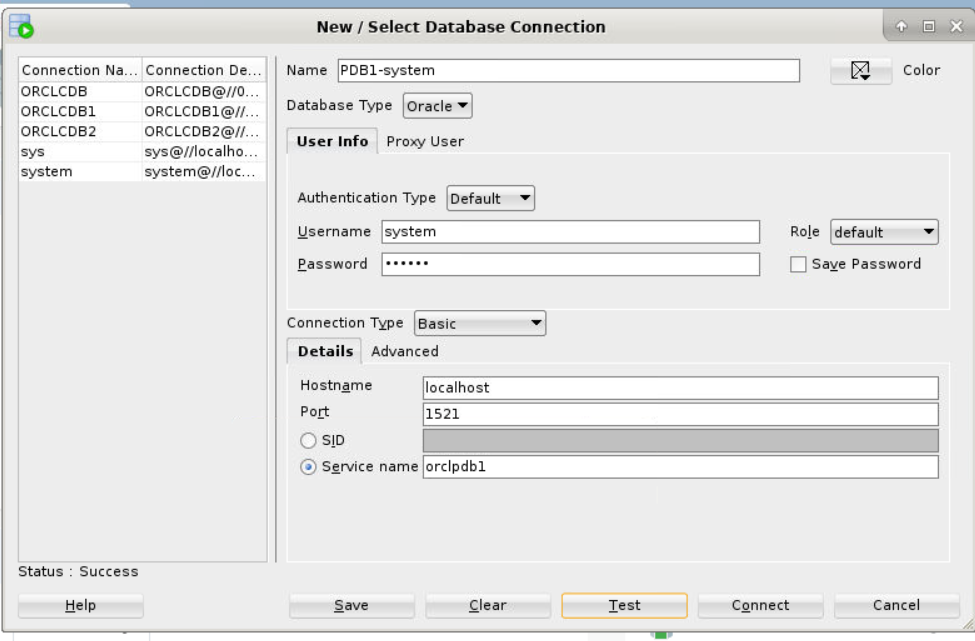
SQL>

1. 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** |  |  |  |  |
|  | …  $ |  |  |  |  |

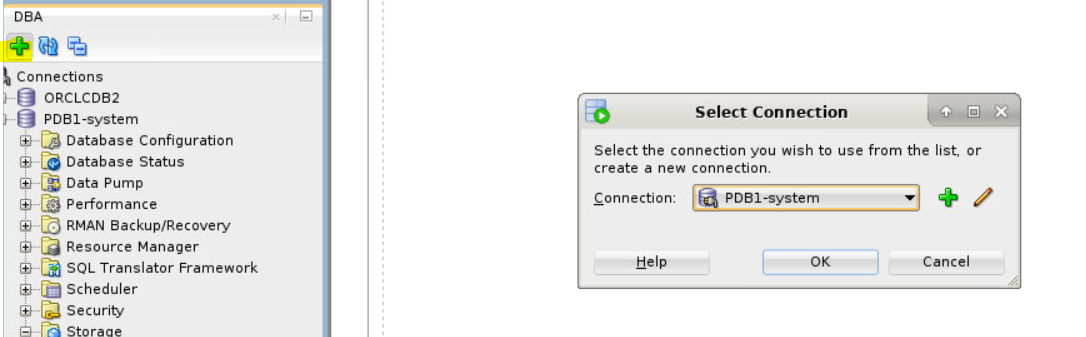
#### Viewing Tablespace Information by SQL\*Developer

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

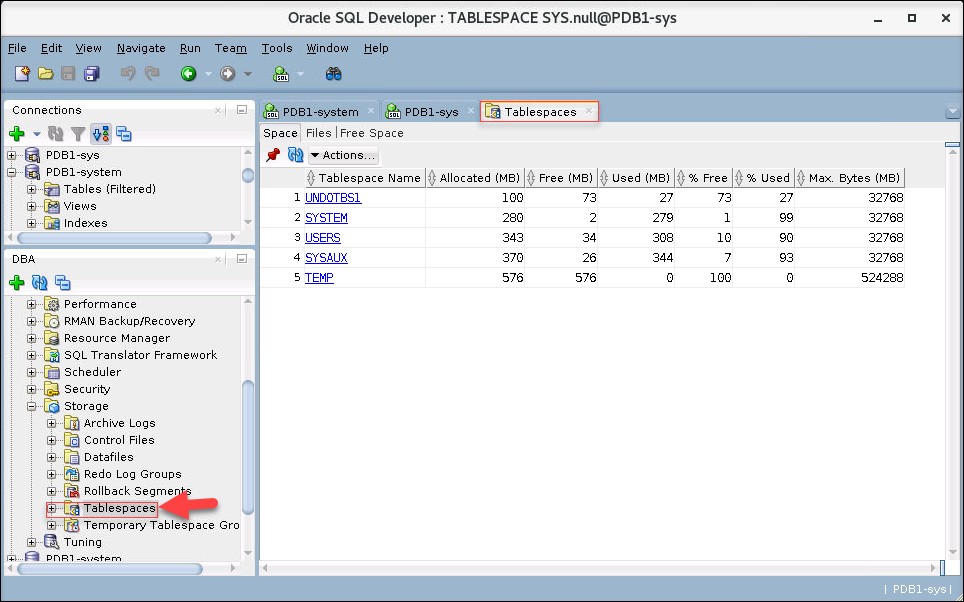


1. Expand **PDB1-system** in the Connections pane.
2. In the DBA view, Expand **PDB1-system**.

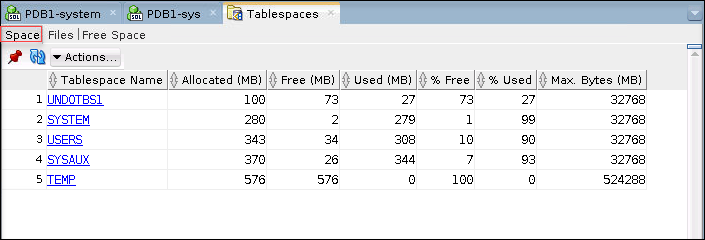
**Note:** If you do not see the DBA panel on the lower left under Connections & Reports panel, then select **View** > **DBA**



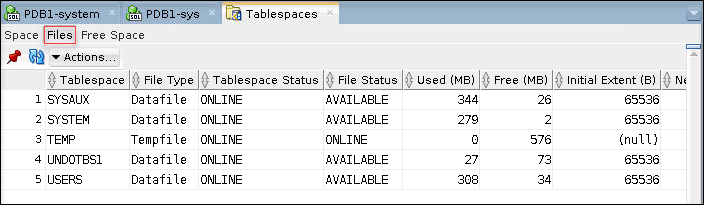
1. Expand **Storage** and then select **Tablespaces**. A new tab named Tablespaces should appear.

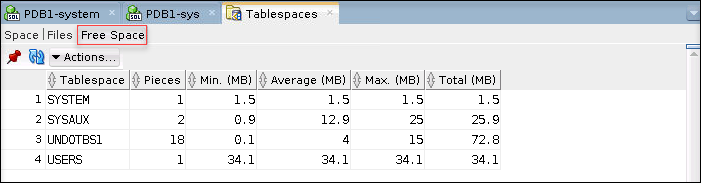


1. All the tablespaces in the **Space** tab are listed with their size, amount of free space, amount used (MB),%Free, %Used, and Maximum Size setting.



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



1. In the **Free Space** tab, Pieces, Min(MB), Average (MB), Max (MB) and Total (MB) are listed.
2. 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.

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

1. 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>

1. Execute the CreateINVENTORYTablespace.sql script.

SQL> **set echo on**

SQL> **@/home/oracle/labs/DBMod\_Storage/CreateINVENTORYTablespace.sql**

SQL> CREATE SMALLFILE TABLESPACE INVENTORY

1. DATAFILE
2. '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF' SIZE 5242880
3. DEFAULT NOCOMPRESS
4. ONLINE
5. SEGMENT SPACE MANAGEMENT AUTO
6. EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

Tablespace created. SQL>

1. Execute the CreateTable\_X.sql script to create and populate the X table. Notice that near the end, you get an error message: unable to extend table PDBADMIN.X by

128 in tablespace INVENTORY. You get this message because the tablespace in which you are trying to create table X is too small. You will remedy this problem in the next section.

SQL> **@/home/oracle/labs/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

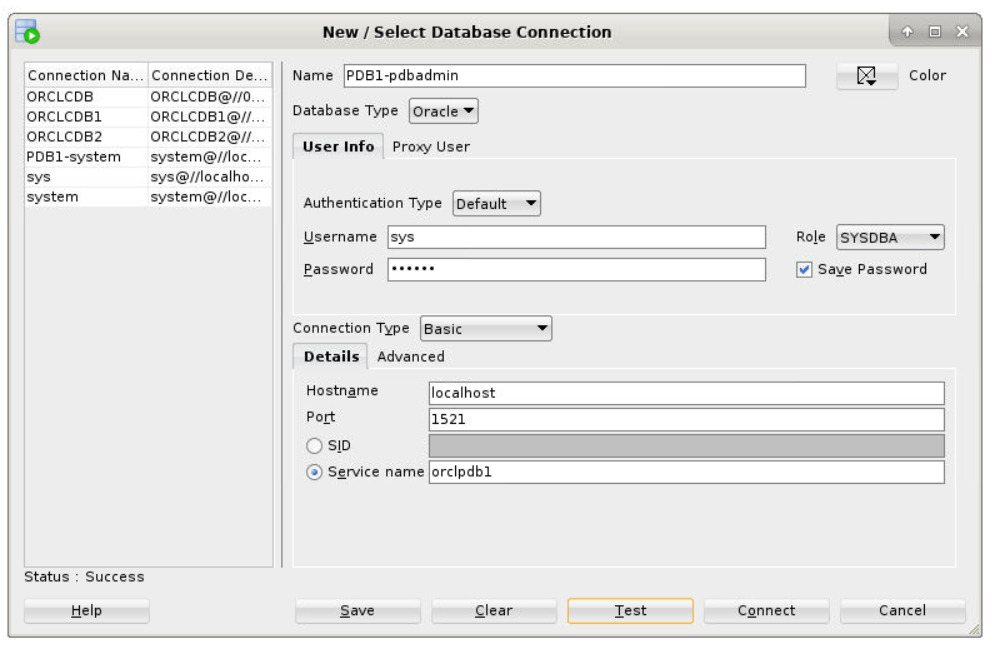
…

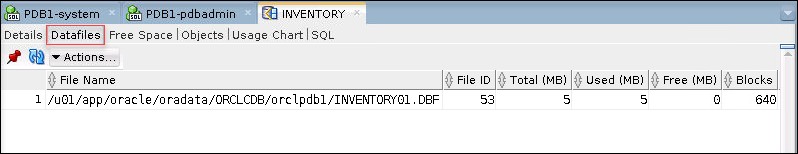
$

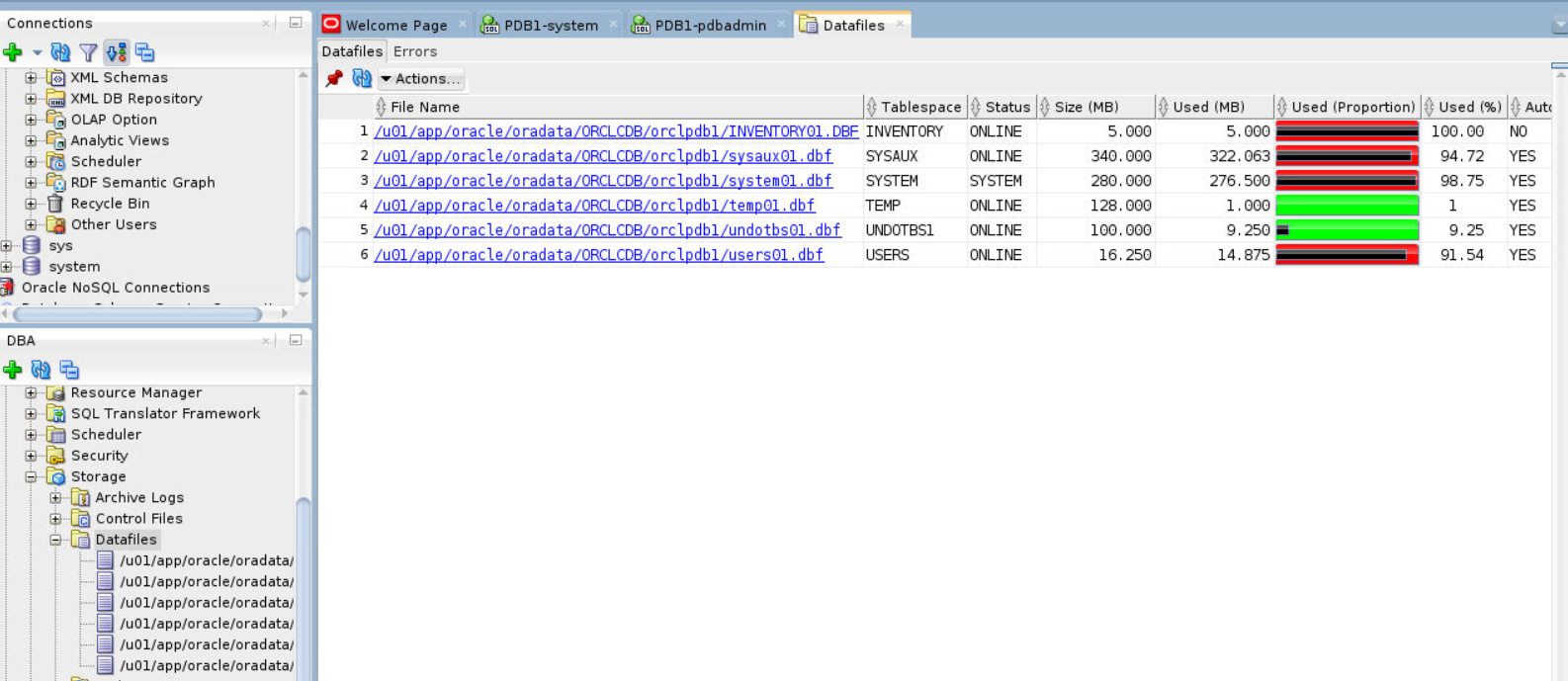
#### Use SQL\*Developer to Increase the Size of the INVENTORY01.DBF Data File

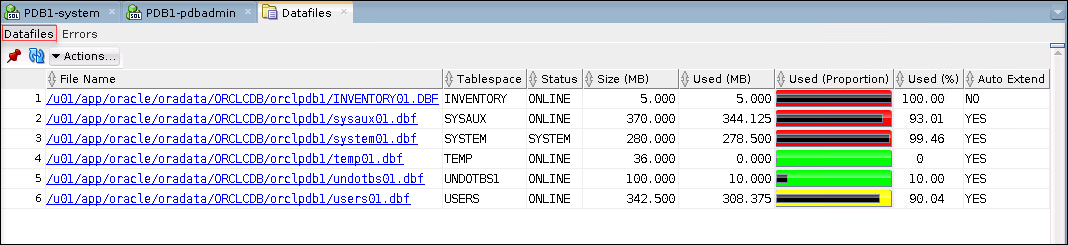
Fix the problem that you encountered in the previous section by increasing the size of the INVENTORY01.dbf data file. Use SQL\*Developer because it provides an easy-to-use interface when working with tablespaces.

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

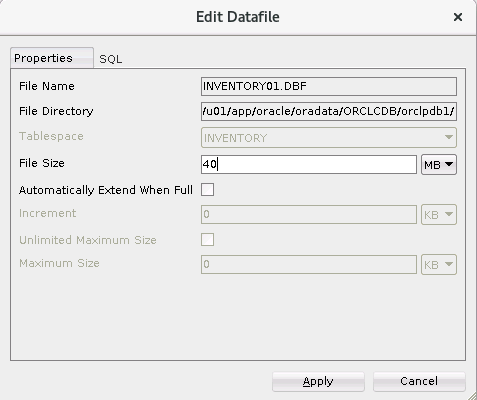


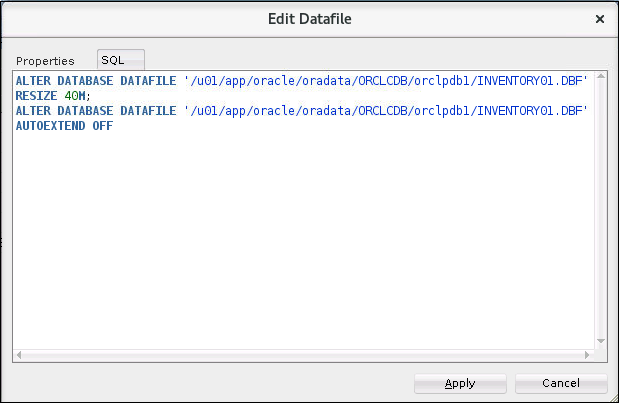
1. In the DBA panel, expand **PDB1-pdbadmin**
2. Expand **Storage** and then select **Tablespaces**.
3. Double click the **INVENTORY** tablespace and select the **Datafiles** tab.
4. Now that you have found the name of **INVENTORY01.DBF** data file.
5. In the DBA pane, select **Datafiles**. You can see that INVENTORY01.DBF datafile is 100% used.

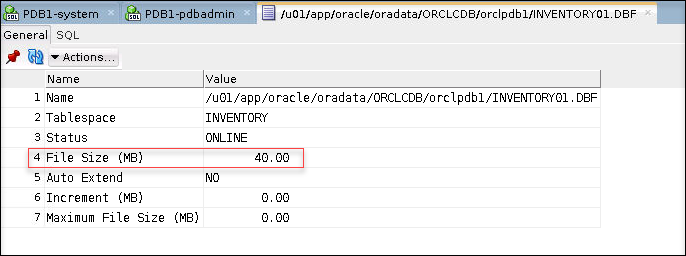




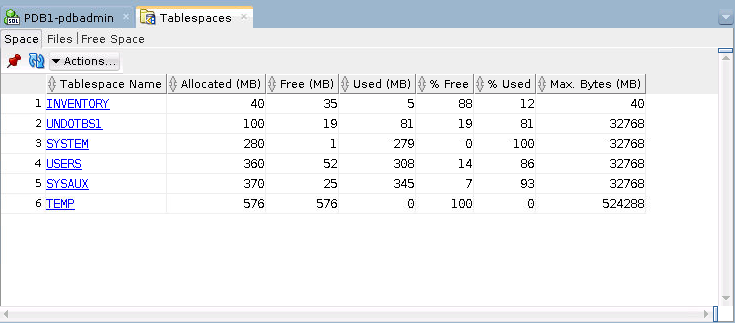
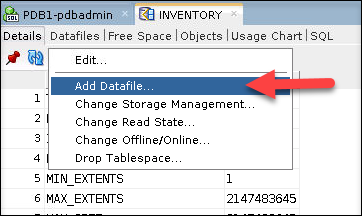
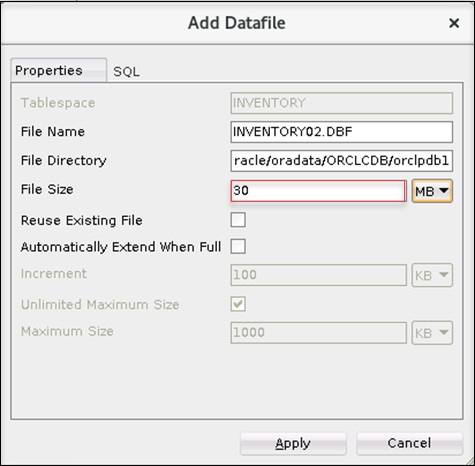
1. Double click the INVENTORY01.DBF filename. Click **Actions > Edit** .
2. In the Edit Datafile box, enter a File Size of **40M**. Don't click Apply just yet.

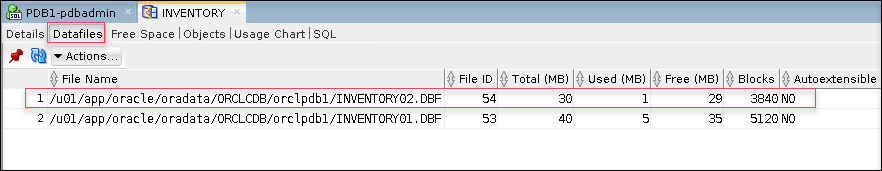


1. Click the **SQL** tab to view the SQL command that performs the resize action.
2. In the dialog box, click **Apply**.
3. In the Successful dialog box, click **OK**. Data file has been successfully resized.
4. Verify that the change is reflected in the SQL\*Developer interface. The size for the

INVENTORY tablespace should now be set to 40MB.

#### Use SQL\*Developer to Add a Data File to the INVENTORY Tablespace

1. In the DBA pane, Expand **Storage**, and click **Tablespaces**
2. Double click the **INVENTORY** tablespace.
3. Expand **Actions** and then select **Add Datafile**.
4. The "Add Datafiles" dialog box is displayed.
   1. Enter File Name: **INVENTORY02.DBF**
   2. Enter File Directory: **/u01/app/oracle/oradata/ORCLCDB/orclpdb1/**
   3. Enter the File Size: **30M**.
5. Click **SQL** tab and view the SQL code being generated.
6. Click **Apply**.
7. In the Successful window, click **OK**
8. Refresh the **INVENTORY** tab by clicking on the Datafiles subtab and verify that it now has two data files: **INVENTORY01.DBF** and **INVENTORY02.DBF**



1. 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.

1. Return to your terminal window.
2. 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>

1. 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

…

$

1. 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>

1. Verify that table X was created in the INVENTORY tablespace.

SQL> **select table\_name from all\_tables where tablespace\_name='INVENTORY';**

TABLE\_NAME X

SQL>

#### Use SQL\*Plus to Drop the INVENTORY Tablespace

1. Drop the INVENTORY tablespace.

SQL> **drop tablespace inventory including contents and datafiles;**

Tablespace dropped. SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. Close the terminal session.

## Practice 16-3: Managing Temporary and Permanent Tablespaces

$ **. 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;**

### 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

1. Run following command as root user: **chmod -R 777 /home/oracle/labs/**
2. 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**

1. View permanent and temporary tablespaces properties in ORCLCDB.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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  15 rows | selected. | 4 |

1. 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';**

TABLESPA CON\_ID

CDATA

1

SQL>

SQL> **select tablespace\_name, con\_id from cdb\_tablespaces where tablespace\_name like 'TEMP%' order by 2;**

TABLESPACE\_NAME CON\_ID

TEMP TEMP

TEMP

1

3

4

SQL>

1. 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>

1. 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>

1. 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>

1. 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>

1. 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>

1. 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/temppdb1\_01.dbf' size 100m;**

Tablespace created. SQL>

1. 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%';**

1. 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>

PROPERTY\_NAME

PROPERTY\_VALUE

DEFAULT\_PERMANENT\_TABLESPACE LDATA DEFAULT\_TEMP\_TABLESPACE TEMP\_PDB1

SQL>

1. 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

1. FROM database\_properties
2. WHERE property\_name LIKE 'DEFAULT\_%TABLE%';

PROPERTY\_NAME

PROPERTY\_VALUE

DEFAULT\_PERMANENT\_TABLESPACE SYSTEM DEFAULT\_TEMP\_TABLESPACE TEMP

SQL>

1. Manage default permanent and temporary tablespaces of users.
   1. 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>

* 1. 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 order** | **username = 'C##U' 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> |  |  |  |

* 1. 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>

* 1. 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>

* 1. Change the temporary tablespace assignment for user LU to MY\_TEMP in ORCLPDB2.

SQL> **alter user lu temporary tablespace my\_temp;**

User altered.

SQL>

* 1. 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>

1. Log out of SQL\*Plus.

SQL> **exit**

…

$

1. 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

$

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

$ **sqlplus system/*password*@orclpdb1**

… SQL>

1. 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>

1. Check the database-wide threshold values for the Tablespace Space Usage metric.
   1. Connect to the root container.

SQL> **alter session set container = cdb$root;**

Session altered.

SQL>

* 1. Query the WARNING\_VALUE and the CRITICAL\_VALUE columns in the DBA\_THRESHOLDS view. The results show that the warning threshold value is 85 and the critical threshold value is 97.

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>

1. 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.
   1. Connect to ORCLPDB1

SQL> **alter session set container = orclpdb1;**

Session altered.

SQL>

* 1. 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

1. DATAFILE '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/tbsalert.dbf'
2. SIZE 120M REUSE LOGGING EXTENT MANAGEMENT LOCAL
3. SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SQL>

1. Query how much free space the TBSALERT tablespace holds by executing the

$HOME/labs/TBSALERT\_free\_space.sql script.

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>

1. Modify the thresholds values for the Tablespace Space Usage metric for the TBSALERT

tablespace. Set the Warning Threshold to 55 and the Critical Threshold to 70.

SQL> **exec dbms\_server\_alert.set\_threshold( - metrics\_id => dbms\_server\_alert.tablespace\_pct\_full,- warning\_operator => dbms\_server\_alert.operator\_ge,- warning\_value => '55',-**

**critical\_operator => dbms\_server\_alert.operator\_ge, - critical\_value => '70', -**

**observation\_period => 1, - consecutive\_occurrences => 1, - instance\_name => 'ORCL', -**

**object\_type => dbms\_server\_alert.object\_type\_tablespace, - object\_name => 'TBSALERT')**

PL/SQL procedure successfully completed. SQL>

1. Verify that the thresholds are set correctly. The query returns a warning value of 55 and a critical value of 70, which indicates that the thresholds are set correctly.

SQL> **select warning\_value, critical\_value from dba\_thresholds where object\_name='TBSALERT';**

WARNING\_VALUE

CRITICAL\_VALUE

55

70

SQL>

1. Query the REASON and RESOLUTION columns from the DBA\_ALERT\_HISTORY view for the

TBSALERT tablespace.

SQL> **col reason format a60**

SQL> **select reason, resolution from dba\_alert\_history where object\_name='TBSALERT';**

REASON

RESOLUT

Threshold is updated on metrics "Tablespace Space Usage" cleared

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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

…

$

1. Check the fullness level of the TBSALERT tablespace to see if the warning level has been reached.
   1. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. Refer *to Course Practice Environment: Security Credentials* for the ***password*** value.

$ **sqlplus system/*password*@orclpdb1**

… SQL>

* 1. Query the size of the TBSALERT tablespace. The results show that the tablespace is 60% full.

SQL> **select sum(bytes) \* 100 / 125829120 from dba\_extents where tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

60

SQL>

* 1. Query the number of free bytes that are left in the TBSALERT tablespace by executing the $HOME/labs/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  SQL> |  | 49283072 |  | 125829120 |  | 39.1666667 |

* 1. 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.

1. Execute and commit the following INSERT statements.

SQL> **INSERT INTO hr.employees4 SELECT \* FROM hr.employees4;**

109568 rows created. SQL> **COMMIT;**

Commit complete.

SQL> **INSERT INTO hr.employees5 SELECT \* FROM hr.employees5;**

109568 rows created. SQL> **COMMIT;**

Commit complete.

SQL>

1. Wait a few minutes.
2. Query the fullness of the tablespace. The result shows that the tablespace is 75% full.

SQL> **SELECT sum(bytes) \* 100 / 125829120**

**FROM dba\_extents**

**WHERE tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

75

SQL>

1. Query the outstanding alerts. The REASON column is updated with a message that states the tablespace is 75 percent full.

If your result still displays 60, wait a little longer and repeat the query. *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>

1. Delete rows from three tables in the HR schema to try to reduce the space used in the tablespace.

SQL> **DELETE hr.employees1;**

219136 rows deleted. SQL> **COMMIT;**

Commit complete.

SQL> **DELETE hr.employees2;**

219136 rows deleted. SQL> **COMMIT;**

Commit complete.

SQL> **DELETE hr.employees3;**

219136 rows deleted SQL> **COMMIT;**

Commit complete.

SQL>

1. Check if there is some reclaimed space after these tables were deleted. The query result indicates that this is not the case. The tablespace is still 75 percent full. Deleting rows frees space in blocks, but it does not return blocks to the tablespace.

SQL> **select sum(bytes) \* 100 / 125829120 from dba\_extents**

**where tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

75

SQL>

#### Create a Segment Advisor Task.

1. Create a Segment Advisor task to get recommendations about the current space situation by executing the **$HOME/labs/seg\_advsr\_task.sql** script.

SQL> **set echo on**

SQL> **@$HOME/labs/DBMod\_Storage/seg\_advsr\_task.sql**

SQL> DECLARE

1. tname VARCHAR2(128) := 'my\_seg\_task';
2. tname\_desc VARCHAR2(128) := 'Get shrink advice for segments in TBSALERT';
3. task\_id NUMBER;
4. object\_id NUMBER;
5. objectname VARCHAR2(100);
6. objecttype VARCHAR2(100);
7. BEGIN
8. dbms\_advisor.create\_task('Segment Advisor', task\_id,tname,tname\_desc,NULL);
9. dbms\_advisor.create\_object(tname,'TABLESPACE','TBSALERT',' ',' ',NULL,' ', object\_id);
10. dbms\_advisor.set\_task\_parameter(tname,'RECOMMEND\_ALL','TRUE');
11. END;

13 /

PL/SQL procedure successfully completed. SQL>

1. Execute the task.

SQL> **DECLARE**

**tname VARCHAR2(128) := 'my\_seg\_task'; BEGIN**

**dbms\_advisor.EXECUTE\_TASK(tname); END;**

**/**

PL/SQL procedure successfully completed. SQL>

1. Query the DBA\_ADVISOR\_TASKS view for recommendations. The recommendation is to get shrink advice for segments stored in the tablespace.

SQL> **SELECT DESCRIPTION FROM dba\_advisor\_tasks WHERE TASK\_NAME='my\_seg\_task';**

DESCRIPTION

Get shrink advice for segments in TBSALERT SQL>

1. Execute the $HOME/labs/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  SQL> | EMPLOYEES5 | The free space in the object is less | than 10MB. |

1. Proceed with the SHRINK operation on the HR.EMPLOYEES1, HR.EMPLOYEES2, and

HR.EMPLOYEES3 tables.

SQL> **ALTER TABLE hr.employees1 SHRINK SPACE;**

Table altered.

SQL> **ALTER TABLE hr.employees2 SHRINK SPACE;**

Table altered.

SQL> **ALTER TABLE hr.employees3 SHRINK SPACE;**

Table altered.

SQL>

1. Check if the SHRINK operations reclaimed unused space by running the following query. The result shows that the tablespace did reclaim unused space. It went down to 30% full from 75% full.

SQL> **SELECT sum(bytes) \* 100 / 125829120**

**FROM dba\_extents**

**WHERE tablespace\_name='TBSALERT';**

SUM(BYTES)\*100/125829120

30.15625

SQL>

1. Drop the TBSALERT tablespace.

SQL> **DROP TABLESPACE tbsalert INCLUDING CONTENTS AND DATAFILES;**

Tablespace dropped. SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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

$

1. 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

…

$

1. 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>

1. Query the compression level of the index created on the **HR.TEST** table. The result indicates that compression is disabled, and therefore, the index is not compressed.

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>

1. Query the space used by the index created on the **HR.TEST** table. The result indicates that 1152 blocks are used.

SQL> **select blocks from user\_segments where segment\_name='I\_TEST';**

BLOCKS

1152

SQL>

1. Exit SQL\*Plus, but keep the terminal window open.

SQL> **exit**

**…**

$

1. View the different compression levels that exist in your Oracle Database version. To do this, use the cat command to review the 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

$

1. 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>

1. Use the Compression Advisor to get recommendations about the space you would save by compressing the index with the COMP\_INDEX\_ADVANCED\_LOW compression level by executing the $HOME/labs/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>

1. Use the Compression Advisor again to get recommendations about the space you would save by compressing the index with the COMP\_INDEX\_ADVANCED\_HIGH compression level by executing the $HOME/labs/DNMod\_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\_uncmp => blkcnt\_uncmp, row\_cmp => row\_cmp,

row\_uncmp => row\_uncmp, cmp\_ratio => cmp\_ratio, comptype\_str => comptype\_str,

subset\_numrows => dbms\_compression.COMP\_RATIO\_MINROWS, objtype => dbms\_compression.OBJTYPE\_INDEX

);

DBMS\_OUTPUT.PUT\_LINE('Block used by compressed index = ' || blkcnt\_cmp);

DBMS\_OUTPUT.PUT\_LINE('Block used by uncompressed index = ' || blkcnt\_uncmp);

DBMS\_OUTPUT.PUT\_LINE('Compression type = ' || comptype\_str); DBMS\_OUTPUT.PUT\_LINE('Compression ratio org = '||cmp\_ratio); END;

/

Block used by compressed index = 130 Block used by uncompressed index = 1029

Compression type = "Compress Advanced High" Compression ratio org = 8

PL/SQL procedure successfully completed.

SQL>

1. Question: Based on the previous steps, which compression ratio is the best—the

COMP\_INDEX\_ADVANCED\_LOW or COMP\_INDEX\_ADVANCED\_HIGH compression level? Answer: The Advanced High Compression ratio (8) is much better than the Advanced Low Compression ratio (1). Therefore, you would be inclined to rebuild the index with Advanced

High Compression.

1. Rebuild the index with Advanced High Compression.

SQL> **alter index hr.i\_test rebuild compress advanced high;**

Index altered.

SQL>

1. Query the compression level of the index created on the HR.TEST table. The result shows that the compression level is ADVANCED HIGH.

SQL> **col index\_name format a20**

SQL> **select index\_name, compression from user\_indexes where index\_name = 'I\_TEST';**

INDEX\_NAME

COMPRESSION

I\_TEST

ADVANCED HIGH

SQL>

1. Query the space used by the index created on the **HR.TEST** table. The space is now 256 blocks.

SQL> **select blocks from user\_segments where segment\_name='I\_TEST';**

BLOCKS

256

SQL>

1. *Question*: Is it possible to revert back to the initial compression level?

*Answer*: Yes.

1. Revert back to the initial compression level.

SQL> **alter index hr.i\_test rebuild nocompress;**

Index altered.

SQL>

1. Query the space used by the index created on the HR.TEST table. The space used is 1152 blocks again.

SQL> **select blocks from user\_segments where segment\_name='I\_TEST';**

BLOCKS

1152

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

## 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>

1. Grant the PDBADMIN usee DBA role.

SQL> **grant dba to pdbadmin;**

Grant succeeded.

SQL>

1. 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>

1. 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>

1. 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

…

$

1. Imagine that the operation in the previous step had lasted 5 hours and that the load had nearly reached its end and other operations were depending on its success.
   1. Question: Are the rows that were inserted into the table lost or definitely inserted? Answer: 2048 rows were inserted.
   2. Question: How could this situation be avoided when you do not know how much space is required for a table to load all its rows?

Answer: In the case of heavy load operations, you can use a corrective action rather than a reactive action after an error is raised. For example, you can use the Resumable Space Allocation feature.

1. Start SQL\*Plus again and connect to ORCLPDB1 as the PDBADMIN user. Refer to *Course Practice Environment: Security Credentials* for the ***password*** value.

$ **sqlplus pdbadmin/*password*@orclpdb1**

… SQL>

1. Enable resumable mode.

SQL> **alter session enable resumable;**

Session altered.

SQL>

1. Re-execute the CreateTable\_X script. The script is suspended.

SQL> **@$HOME/labs/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>

1. Question: Why is the script suspended?

Answer: Enabling the resumable mode for your session suspends the failing statement during 7200 seconds (2 hours), by default.

1. Question: Is there any warning message to tell you the load is suspended? Answer: No. If the script does not execute any further, check the alert log file or the

DBA\_RESUMABLE view. An operation-suspended alert is issued on the object that needs

allocation of resource for the operation to complete.

#### Window 2: Resolve a Suspended Script

1. Open another terminal window. This will be referred to as Window 2.
2. Source the oraenv script.

$ **. oraenv**

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

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

$

1. 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>

1. Query the DBA\_RESUMABLE view for information about the suspended script. The DBA\_RESUMABLE view lists all resumable statements executed in the system. Your times and session information will be different from those shown below.

SQL> **set 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>

1. Exit SQL\*Plus, but keep the terminal window open.

SQL> **exit**

…

$

1. Check the alert log file for information about the suspended script. The log states that the suspension occurred because the table could not be extended.

$ **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

$

1. Proceed with the appropriate corrective action. Because the INVENTORY tablespace is not autoextensible, you can configure it as autoextensible with a size limit.
   1. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. . Refer to *Course Practice Environment: Security Credentials* for the ***password*** value.

$ **sqlplus system/*password*@orclpdb1**

… SQL>

* 1. Query the DBA\_DATA\_FILES view to verify whether the INVENTORY tablespace is autoextensible. The result shows that the tablespace is not.

SQL> **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>

* 1. 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>

* 1. 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

1. Return to Window 1. Notice that the session is no longer suspended. The results show that 2048 rows were created, and the transaction was committed. After the resource had been allocated, the operation completed, and the operation-suspended alert cleared.

SQL> INSERT INTO x

2 SELECT \* FROM x ; 2048 rows created. SQL> COMMIT;

Commit complete.

SQL> quit

…

$

1. Close the terminal window.

#### Window 2: Verify that there are no Suspended Sessions.

1. Return to Window 2. Verify that there are no suspended sessions in the system by querying the DBA\_RESUMABLE view again.

SQL> **select status, name, sql\_text, error\_msg from dba\_resumable;**

no rows selected SQL>

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

SQL> **exit**

...

$ **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 = [orclcdb] ? **orclcdb**

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

$ **sqlplus system**

Enter password: ***password***

connected to:

… SQL>

1. 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**

SQL>

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

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.

1. 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>

1. Exit from SQL\*Plus.

SQL> **exit**

…

$

1. 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**

…

1. 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

1. 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**

…

$

1. 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

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

privilege. This method of connecting uses OS authentication.

$ **sqlplus / as sysdba**

… SQL>

1. Create a common user named c##CDB\_ADMIN1 by using the CREATE USER command. Set the USERS tablespace as the default and TEMP as the temporary tablespace. Also, unlock the account so that c##CDB\_ADMIN1 can log in right away.

**Important**! To create a common user, you must start the user name with c## or C##, and you must include the CONTAINER=ALL clause so that the user's identity and password are created in all the containers.

**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>

1. Grant c##CDB\_ADMIN1 the DBA role, the CREATE SESSION privilege, and the SYSDBA

privilege in all containers. This is an example of granting privileges and a role commonly.

SQL> **grant create session, dba, sysdba to c##cdb\_admin1 container=all;**

Grant succeeded.

SQL>

1. **Question:** Can you use the following statement to complete the same operation (granting privileges and a role commonly)?

GRANT create session, dba TO c##CDB\_ADMIN1;

**Answer:** No, because without the CONTAINER=ALL clause, the CREATE SESSION privilege and DBA role are granted locally (in the root container only) to c##CDB\_ADMIN1, and not to each c##CDB\_ADMIN1 user in each PDB.

1. List the common users by querying the DBA\_USERS view. Scroll down and verify that

c##CDB\_ADMIN1 is included.

SQL> **select distinct username from dba\_users where common='YES' order by username;**

USERNAME

ANONYMOUS APPQOSSYS AUDSYS C##CDB\_ADMIN1 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.

1. Disconnect from the root container.

SQL> **disconnect**

… SQL>

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

SQL> **show user** USER is "" SQL>

1. Connect to the root container as *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>

1. Show the current container name.

SQL> **show con\_name**

CON\_NAME

CDB$ROOT SQL>

1. Show the current user. The current user is SYS, which means the c##CDB\_ADMIN1 user can now do anything that the SYS user can do.

**Note:** Audit trails will show the / user with the SYSDBA privilege, not SYS.

SQL> **show user**

USER is "SYS" SQL>

1. View the list of privileges for the c##CDB\_ADMIN1 user by querying the SESSION\_PRIVS

static data dictionary view. Scroll down to view the privileges listed.

SQL> **select \* from session\_privs order by privilege;**

PRIVILEGE

ADMINISTER ANY SQL TUNING SET ADMINISTER DATABASE TRIGGER

…

USE ANY SQL TRANSLATION PROFILE

WRITE ANY ANALYTIC VIEW CACHE

253 rows selected.

SQL>

1. Disconnect from the root container.

SQL> **disconnect**

… SQL>

1. Connect to the root container as c##CDB\_ADMIN1 again, but this time, do not exercise the SYSDBA privilege. Refer to *Course Practice Environment: Security Credentials* for the ***password*** value.

SQL> **connect c##cdb\_admin1/*password***

Connected.

SQL>

1. Show the current user. You are connected as c##CDB\_ADMIN1. Because you included the CONTAINER=ALL clause when granting the CREATE SESSION privilege and DBA role, c##CDB\_ADMIN1 can connect as a regular user to any open PDB and perform system and object operations that the DBA role allows.

SQL> **show user**

USER is "C##CDB\_ADMIN1" SQL>

1. View the list of privileges for the c##CDB\_ADMIN1 user by querying the SESSION\_PRIVS static data dictionary view. Scroll through the list of privileges. Notice that there are fewer privileges listed than when c##CDB\_ADMIN1 was connected with the SYSDBA privilege.

SQL> **select \* from session\_privs order by privilege;**

PRIVILEGE

ADMINISTER ANY SQL TUNING SET ADMINISTER DATABASE TRIGGER ADMINISTER RESOURCE MANAGER

…

UPDATE ANY TABLE

USE ANY JOB RESOURCE

USE ANY SQL TRANSLATION PROFILE

237 rows selected.

SQL>

1. Switch to ORCLPDB1 by issuing the ALTER SESSION command.

SQL> **alter session set container = orclpdb1;**

Session altered.

SQL>

1. 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.

1. 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>

1. 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>

1. 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>

1. Disconnect c##CDB\_ADMIN1 from PDB1.

SQL> **disconnect**

… SQL>

1. 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>

1. Show the current user. You are connected as PDB1\_ADMIN1.

SQL> **show user**

USER is "PDB1\_ADMIN" SQL>

1. 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>

1. 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>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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>

1. Create a local user account named INVENTORY. Set the default tablespace to the USERS tablespace and grant unlimited quota on that tablespace. Refer to *Course Practice Environment: Security Credentials* for the ***password*** value.

SQL> **create user inventory identified by *password***

**default tablespace users quota unlimited on users;**

User created. SQL>

1. Grant the CREATE SESSION privilege to INVENTORY.

SQL> **grant create session to inventory;**

Grant succeeded.

SQL>

1. List the local user accounts for 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

1. Disconnect PDB1\_ADMIN from ORCLPDB1.

SQL> **disconnect**

… SQL>

1. 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>

1. List the privileges for INVENTORY by querying the SESSION\_PRIVS view. The results show that INVENTORY has the CREATE SESSION privilege.

SQL> **select \* from session\_privs order by 1;**

PRIVILEGE

CREATE SESSION SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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

$

1. To find out the user that you are currently logged in as, execute whoami. The result shows that you are currently logged in to the OS as the oracle user.

$ **whoami**

oracle

$

1. Find out more about the oracle user. For example, verify that oracle is part of the dba

group.

* 1. The /etc/passwd file is a text file that lists user account information needed for logging in to the OS. Execute the following command to search for the oracle user. The format of the row is user:password:user ID:primary group ID:home directory: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

$

* 1. The information above tells you that oracle's primary group ID is 54321. To find the name of that group, search for it in the group file. The result shows that the oracle user's primary group is the oinstall group.

$ **grep oinstall /etc/group**

oinstall:x:54321:

$

* 1. Investigate further. Search for oracle in the group file. The results tell you that oracle is a user in the dba group. The dba group is the Database Administrator Group, and any user in this group has the SYSDBA system privilege. So, if you log on to the Oracle database by using OS authentication and exercise the SYSDBA privilege, then the oracle user becomes the SYS user. If you recall, to log on using OS authentication, all you need to specify is CONNECT / AS SYSDBA. The / tells SQL\*Plus to look up the privileges for the OS user's group.

$ **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

$

* 1. 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),5 4325(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$PWFILE\_USERS view contains information about users that have been granted administrative privileges.

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

privilege.

$ **. oraenv**

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

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

$ **sqlplus / as sysdba**

… SQL>

1. View the columns in the V$PWFILE\_USERS view by issuing the DESCRIBE command.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL> d**escribe v$pwfile\_users** | | | | |
| Name |  | Null? |  | Type |
| USERNAME SYSDBA SYSOPER SYSASM SYSBACKUP SYSDG SYSKM  ACCOUNT\_STATUS PASSWORD\_PROFILE LAST\_LOGIN LOCK\_DATE EXPIRY\_DATE EXTERNAL\_NAME AUTHENTICATION\_TYPE COMMON  CON\_ID |  |  |  | VARCHAR2(128) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(5) VARCHAR2(30) VARCHAR2(128)  TIMESTAMP(9)WITH TIME ZONE DATE  DATE VARCHAR2(1024) VARCHAR2(8) VARCHAR2(3) NUMBER |
| SQL> | | | | |

1. List the users in the password file by querying the V$PWFILE\_USERS view.

SQL> **col username format a20**

SQL> **select username from v$pwfile\_users;**

USERNAME SYS

C##CDB\_ADMIN1

SQL>

1. Find out the SYS user's account status and whether the SYS user has the SYSDBA privilege by querying the V$PWFILE\_USERS view. ACCOUNT\_STATUS shows if the administrative user is OPEN, LOCKED (the user can no longer connect), or EXPIRED (the user must change the password at the connection).

SQL> **select account\_status, sysdba from v$pwfile\_users where username='SYS';**

ACCOUNT\_STATUS

SYSDB

OPEN

TRUE

SQL>

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

SQL> **exit**

…

$

1. 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>

1. List the system privileges granted to the PDBADMIN user by querying the DBA\_SYS\_PRIVS view. This view describes system privileges granted to users and roles. The results show that 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>

1. 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>

1. List the system privileges granted to the PDB\_DBA role by querying the ROLE\_SYS\_PRIVS

view.

* 1. Query the ROLE\_SYS\_PRIVS view. This view describes system privileges granted to roles. Information is provided only about roles to which the user has access. Because you're connected to 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>

1. List the roles that are granted to the PDB\_DBA role by querying the DBA\_ROLE\_PRIVS view. The results show that the PDB\_DBA role is granted the CONNECT role.

SQL> **col granted\_role format a15**

SQL> **select granted\_role from dba\_role\_privs where grantee = 'PDB\_DBA' order by 1;**

GRANTED\_ROLE CONNECT

SQL>

1. List the privileges granted to the CONNECT role by querying the ROLE\_SYS\_PRIVS view. The results show that the CONNECT role consists of the SET CONTAINER and CREATE SESSION privileges.

SQL> **select privilege from role\_sys\_privs where role='CONNECT' ORDER BY 1;**

PRIVILEGE

CREATE SESSION SET CONTAINER

SQL>

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

#### Grant the DBA Role to PDBADMIN

1. Grant the DBA role locally to PDBADMIN.

SQL> **grant dba to pdbadmin;**

Grant succeeded.

SQL>

1. List the roles that are granted to PDBADMIN by querying the DBA\_ROLE\_PRIVS view. The results show that PDBADMIN is now granted the DBA and PDB\_DBA roles.

SQL> **select granted\_role from dba\_role\_privs where grantee = 'PDBADMIN' order by 1;**

GRANTED\_RO

DBA PDB\_DBA

SQL>

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

SQL> **exit**

…

$

1. Close all terminals

## Practice 20-2: Using SQL\*Developer to Create Local Roles

### Overview

In this practice, the PDBAMIN 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

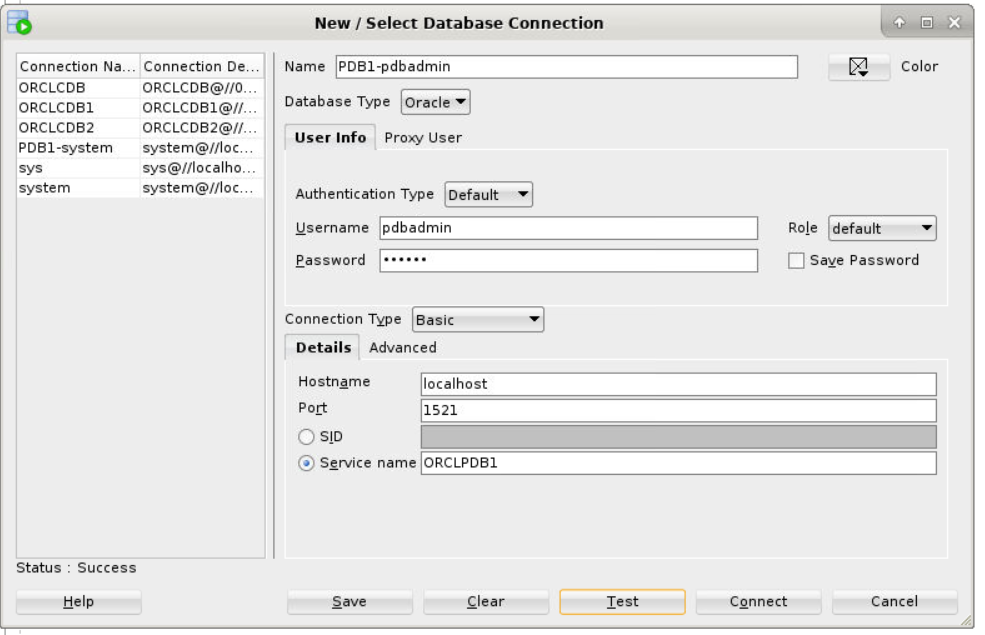
You are currently logged in as the oracle OS user.

You completed Practice 2-1 Granting the DBA Role to PDBADMIN.

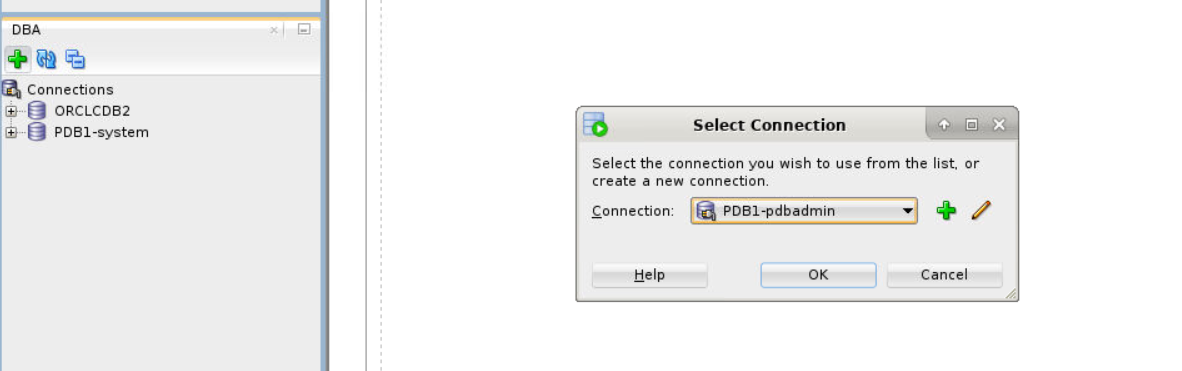
### Tasks

#### Log in to SQL\*Developer (PDB1)

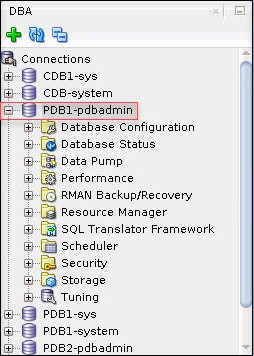
1. Create a connection to ORCLPDB1 as the PDBADMIN.
   1. Start SQL\*Developer. The SQL\*Developer icon is on the desktop.

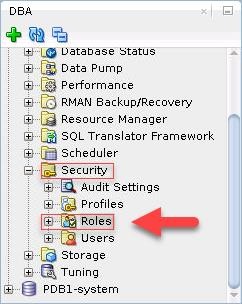


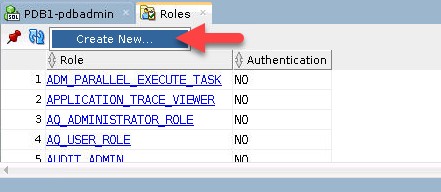
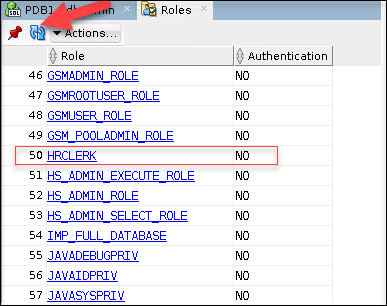
* 1. In the Connections pane on the left, click the Name: **PDB1-pdbadmin** (a connection for ORCLPDB1 as PDBADMIN)
  2. In the DBA connection box, click the connection: **PDB1-pdbadmin**



#### Create the HRCLERK Role

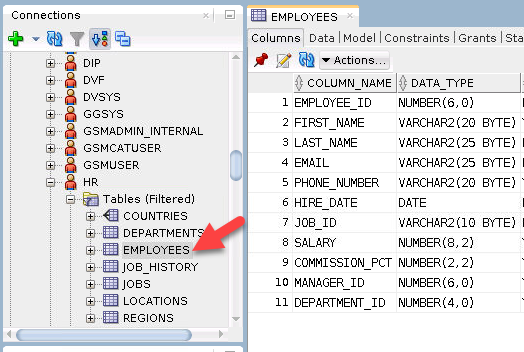
1. Expand the PDB1-pdbadmin connection in the DBA box.
2. Expand **Security** and then select **Roles**.



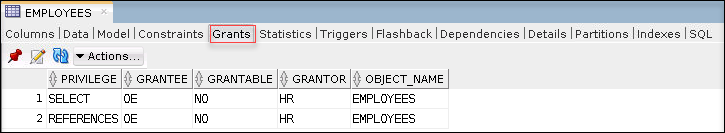
1. In the Roles tab, select **Actions > Create New …**
2. In the Create Role box,
   1. On the **Roles** tab, enter Role Name: **HRCLERK**.
   2. Click the **SQL** tab to view the SQL statement.
   3. Click **Apply**.
   4. In the Successful box, click **OK**.
   5. In the Create Role box, click **Close**.
3. Verify that the **HRCLERK** role is listed in the table. Note you may have to click on refresh

#### Add Object Privileges to HRCLERK role

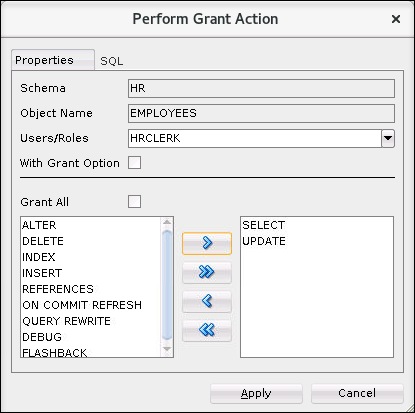
1. In the Connections box, expand **PDB1-pdbadmin> Other USERS**> **HR** > **Tables,**

then click **Employees.**

1. In the **EMPLOYEES** tab on the right, click the **Grants** subtab.

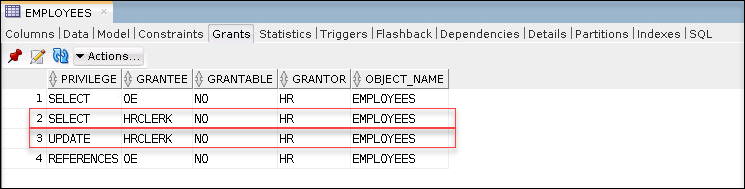


#### Click Actions > Privileges > Grant.

* 1. The Perform Grant Action box is displayed.
  2. In the Users/Roles drop-down list, select **HRCLERK**.
  3. In the left hand list, click **SELECT** and move it to right hand list.
  4. In the left hand list, click **UPDATE** and move it to right hand list.
  5. Click the SQL tab to view the SQL statement.

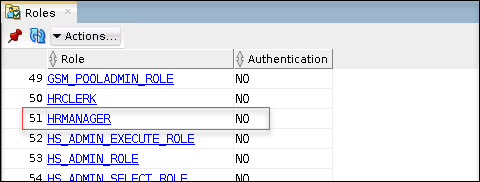
grant UPDATE, SELECT on "HR"."EMPLOYEES" to "HRCLERK" ;

* 1. Click **Apply.**
  2. In the Successful box click **OK.**
  3. Refresh on the Grants subtab and verify SELECT and UPDATE on HR.EMPLOYEES

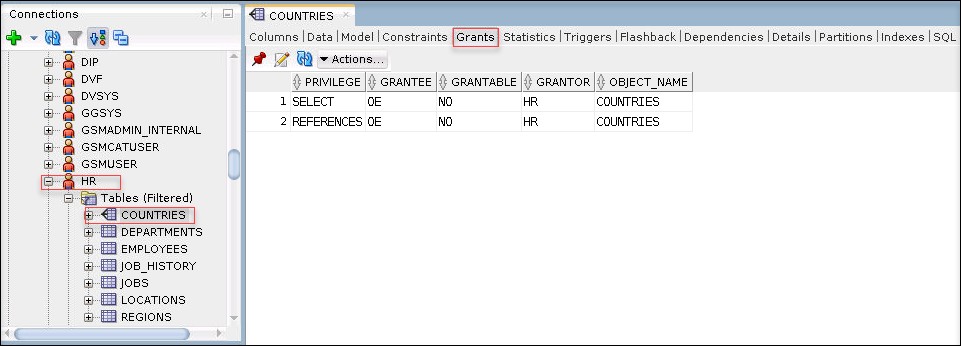
have been granted to HRCLERK.

#### Create the HRMANAGER Role

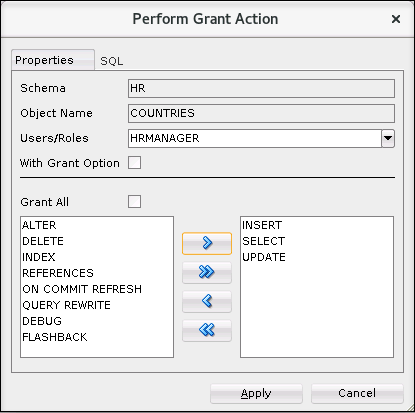
The steps in this section are similar to the ones in the previous section.

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,
   1. On the Roles tab, enter Role Name: **HRMANAGER**.
   2. Click the SQL tab to view the SQL statement.
   3. Click **Apply**.
   4. In the Successful box, click **OK**.
   5. In the Create Role box, click **Close**.
5. Verify that the **HRMANAGER** role is listed in the table.

#### Add Object Privileges to HRMANAGER role

1. In the Connections box, expand **Other USERS**> **HR> Tables,** and click **COUNTRIES.**
2. Click the **Grants** subtab.

#### Click Actions > Privileges > Grant.

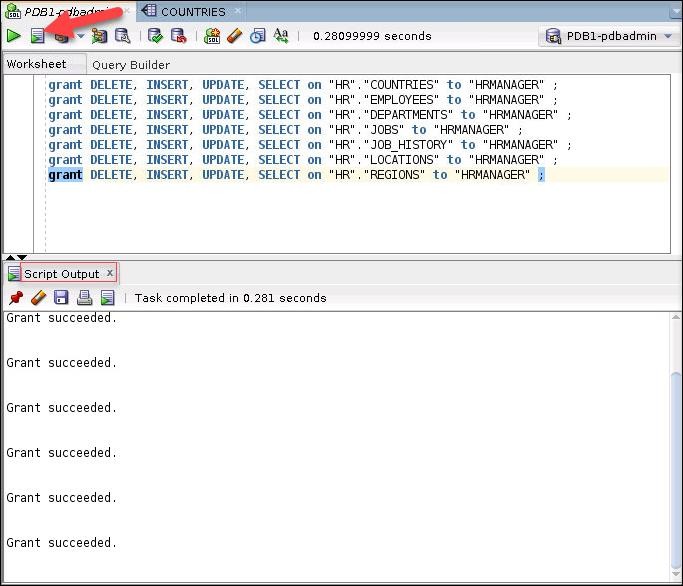
* 1. The Perform Grant Action box is displayed.
  2. In the Users/Roles drop-down list, select **HRMANAGER**.
  3. In the left hand list, click **DELETE** and move it to right hand list.
  4. Repeat for **INSERT**,**SELECT**, and **UPDATE**.
  5. Click the SQL tab to view the SQL statement.

grant DELETE, INSERT, UPDATE, SELECT on "HR"."COUNTRIES" to "HRMANAGER" ;

* 1. Highlight the SQL statement, and copy it (use the middle mouse button or ctrl-V).
  2. Cancel the Perform Grant Action box, click **Cancel**.
  3. 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**
  4. Paste the SQL Statement into the worksheet (use the middle mouse button or ctrl-V).
  5. Press **Crtl-Enter** or the Green right arrow  to execute the statement.
  6. 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" ;

* 1. 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:



1. Verify the HRMANAGER role and required privileges.
   1. In the DBA box, expand **Security** and click **Roles**.
   2. Double-click the **HRMANAGER** role in the view on the right.
   3. In the **HRMANAGER** tab, click the **Object Privs** subtab.
   4. Verify the privileges granted on the HR tables are shown.
2. 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

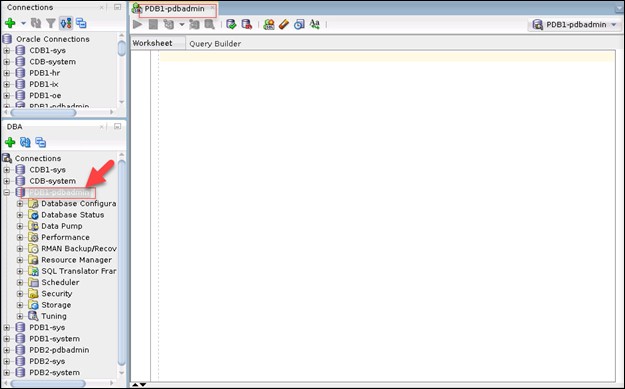
A local profile is a profile that resides in a single PDB. Therefore, to create one, you must log in to the PDB.

### Assumptions

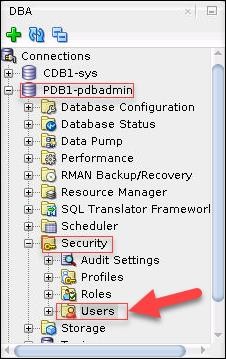
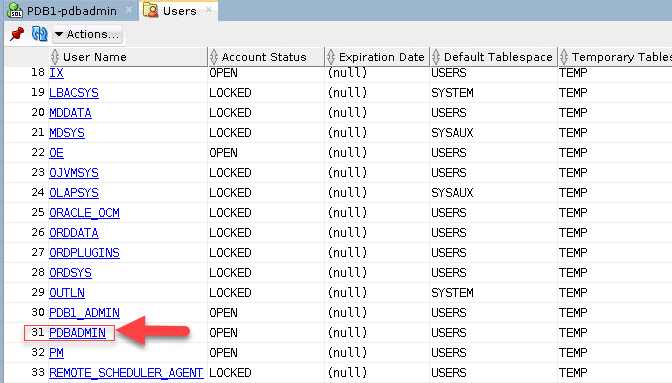
You are currently logged in as the oracle OS user. The PDBADMIN user has been granted the DBA role.

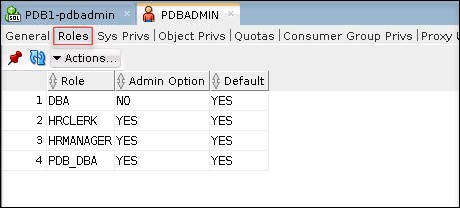
### Tasks

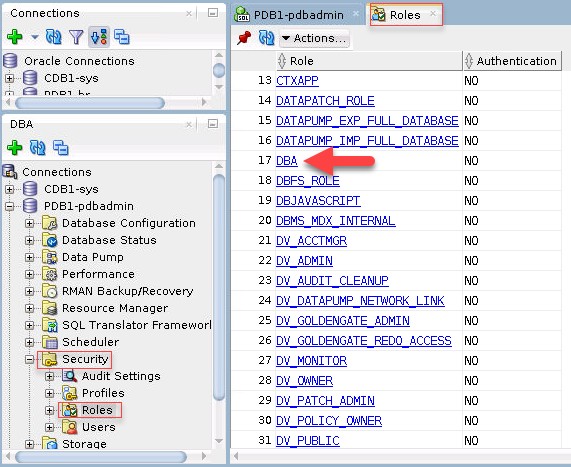
#### Log in to SQL\*Developer (ORCLPDB1)

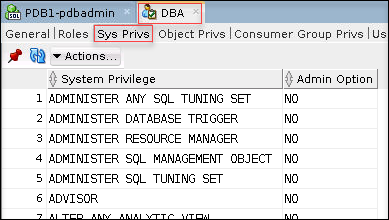
1. Launch SQL\*Developer.
2. Click Menu View > DBA
   1. In the DBA box, double click: **PDB1-pdbadmin.**
   2. A worksheet for PDB1-pdbadmin will appear on the right.

#### View Privileges and Roles for PDBADMIN

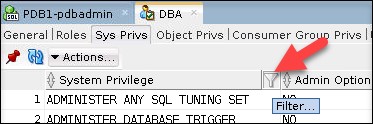
1. In the DBA box, expand **PDB1-pdadmin** > **Security**> **Users**
2. On the Users tab, select **PDBADMIN**
3. The PDBADMIN tab is displayed. Click the **Roles** subtab. Notice that DBA is a ROLE.

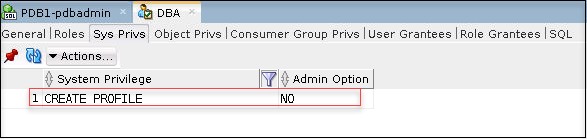
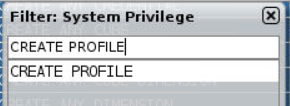


1. In the DBA box, expand **Security** and **Roles** and select **DBA**.
2. In the DBA tab, click the **Sys Privs** subtab. Scroll down through the list of privileges.



1. 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.**

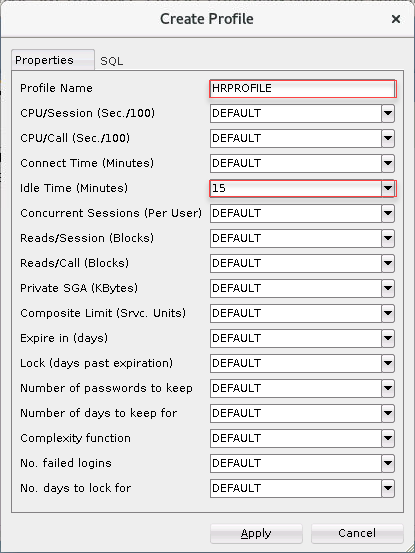


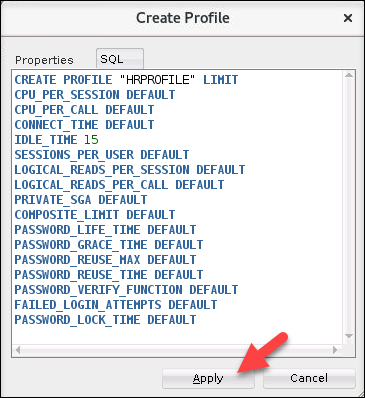


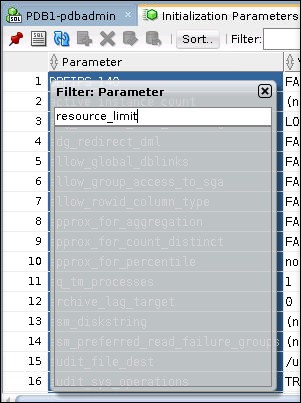
1. Close the DBA Role tab.

#### Create a Local Profile

1. In the DBA box, expand **Security** and click **Profiles**.
2. In the Profiles tab, click **Actions** > **Create New …**
3. In the Create Profile box, enter:
   1. Profile Name: **HRPROFILE**
   2. IdleTime (minutes): **15**

Leave all other fields set to Default.

1. Click the **SQL** tab to review the SQL command for this task. When done, click **APPLY.**
2. In the Successful window, click **OK**.
3. Verify that HRPROFILE is in the list of profiles. Note, you may need to click **Refresh ** **Set the RESOURCE\_LIMIT Initialization Parameter**
4. In the DBA box, expand **Database Configuration** and select **Initialization Parameters**.
5. Hoover over the Parameter header and click the Filter icon. Type **resource\_limit** in the Filter: Parameter field and press **Enter**.



1. The RESOURCE\_LIMIT initialization parameter is listed in the table. Verify that the

RESOURCE LIMIT value is set to **TRUE**.

1. If RESOURCE\_LIMIT is not set to true, perform the following steps:
   1. Double-click the Value field of the RESOURCE\_LIMIT row.
   2. Select **TRUE**, and click somewhere outside the Value field.
   3. The row number will have an asterisk, then you can press F11 or click the  icon to commit the change,
   4. In the Commit Strategy dialog box, check both **Memory** and **SPFile** boxes and then click **Apply**.
2. 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.

1. Open a new terminal window and source the oraenv script for the **orclcdb** database. .

$ **. oraenv**

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

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

$

1. Connect to ORCLPDB1 as the local DBA, PDBADMIN. Refer to *Course Practice Environment: Security Credentials* for the ***password*** value.

$ **sqlplus pdbadmin/*password*@orclpdb1**

… SQL>

1. Issue the ALTER PROFILE command to set the INACTIVE\_ACCOUNT\_TIME limit in the profile to 10 days.

SQL> **alter profile hrprofile limit inactive\_account\_time 10;**

ALTER PROFILE hrprofile LIMIT INACTIVE\_ACCOUNT\_TIME 10

\*

ERROR at line 1:

ORA-02377: invalid profile limit INACTIVE\_ACCOUNT\_TIME

SQL>

1. Question: Is INACTIVE\_ACCOUNT\_TIME a valid profile limit? To find out, query the

DBA\_PROFILES view and confirm that INACTIVE\_ACCOUNT\_TIME is listed in the table.

#### 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.

1. 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

//

//

is allowed

//

limit cannot be less than 15 days.

\* For the PASSWORD\_GRACE\_TIME profile parameter, 0

as a permissible value.

SQL>

1. 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>

1. 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 KERNEL CONNECT\_TIME KERNEL CPU\_PER\_CALL KERNEL CPU\_PER\_SESSION KERNEL IDLE\_TIME

KERNEL LOGICAL\_READS\_PER\_CALL

DEFAULT DEFAULT DEFAULT DEFAULT 15

DEFAULT

KERNEL LOGICAL\_READS\_PER\_SESSION KERNEL PRIVATE\_SGA

KERNEL SESSIONS\_PER\_USER PASSWORD FAILED\_LOGIN\_ATTEMPTS PASSWORD INACTIVE\_ACCOUNT\_TIME PASSWORD PASSWORD\_GRACE\_TIME PASSWORD PASSWORD\_LIFE\_TIME PASSWORD PASSWORD\_LOCK\_TIME PASSWORD PASSWORD\_REUSE\_MAX PASSWORD PASSWORD\_REUSE\_TIME

PASSWORD PASSWORD\_VERIFY\_FUNCTION

DEFAULT DEFAULT DEFAULT DEFAULT 15 DEFAULT DEFAULT DEFAULT DEFAULT DEFAULT

DEFAULT

17 rows selected.

SQL>

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

SQL> **exit**

…

$ **exit**

1. Question: What will a DBA have to do if a database user account gets locked due to this new limit?

Answer: The DBA will have to unlock the database user account to make it available for use again by issuing the following command:

ALTER USER acct\_user IDENTIFIED BY *password* ACCOUNT UNLOCK;

## Practice 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 HRPROFILE to the accounts as well as various privileges and roles that you've already created in previous practices. Afterward, test the accounts by logging in to SQL\*Plus as each user. Also, test the idle time setting in HRPROFILE.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **User Account** | **Description** | **Privileges/Roles** | **Method to Use to Create User** |
| Jenny Goodman | JGOODMAN | A new HR manager | CREATE SESSION  privilege | SQL\*Developer |
|  |  |  | HRCLERK role |  |
|  |  |  | HRMANAGER role |  |
| David Hamby | DHAMBY | A new HR clerk | CREATE SESSION  privilege  HRCLERK role | SQL\*Developer |
| Rachel Pandya | RPANDYA | A new HR clerk | CREATE SESSION  privilege  HRCLERK role | SQL script with substitution variables |

### Assumptions

You are currently logged in as the oracle user.

You created a local profile in ORCLPDB1 named HRPROFILE 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

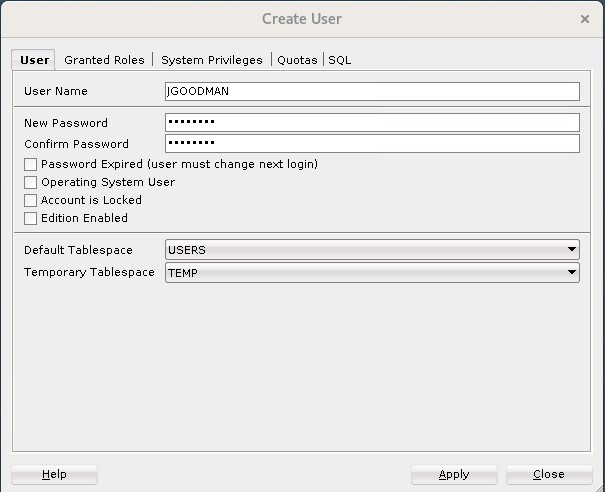
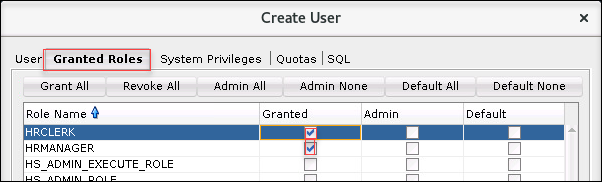
### Tasks

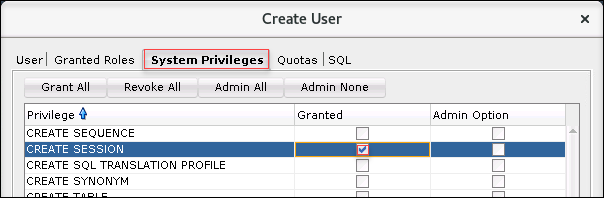
#### Connect to ORCLPDB1 as PDBADMIN Using SQL\*Developer

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.

1. In the DBA box, expand **PDB1-pdbadmin**.
2. Expand **Security** and select **Users.**
3. Click **Actions** > **Create New ..**.
4. 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
5. On the Granted Roles tab, check the Granted box for **HRCLERK** role, and **HRMANAGER** role.
6. On the System Privileges tab, check the Granted box for **Create Session** privilege.



1. 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.

1. In the Successful dialog box, click **OK**.
2. 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.

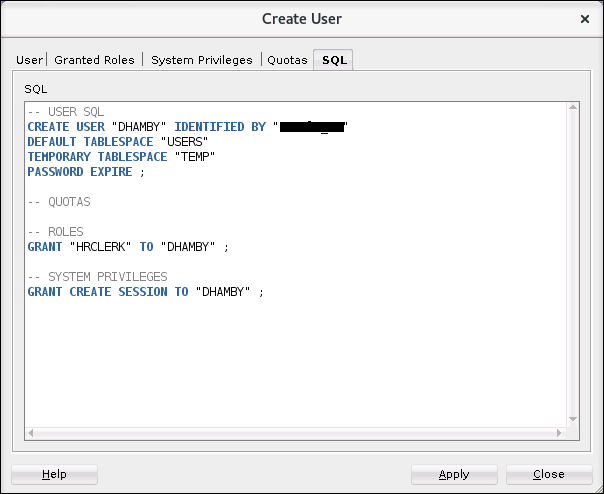
#### Create a User Account for David Hamby

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.

1. If the Users tab is not open, expand **Security** and then select **Users**.
2. Click **Actions > Create New ..**.
3. 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**

1. On the Granted Roles tab, select **HRCLERK** role and check **Granted**.
2. On the System Privileges tab, select the **CREATE SESSION** privilege, and check **Granted**.
3. Click **SQL** tab.
4. 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.

1. Copy the SQL statements in the previous step to the clipboard.
2. Click **Apply** to create the DHAMBY user.
3. Click **OK** in the Successful box.
4. Click the **PDB1-pdadmin** tab to view the worksheet.
5. **Paste** the SQL statements in the worksheet. Change every occurrence as follows:

|  |  |
| --- | --- |
| **Item To Change** | **Change To** |
| **DHAMBY** | **&&username** |
| **oracle\_4U** | **&&password** |
| **HRCLERK** | **&&role** |

1. 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" ;

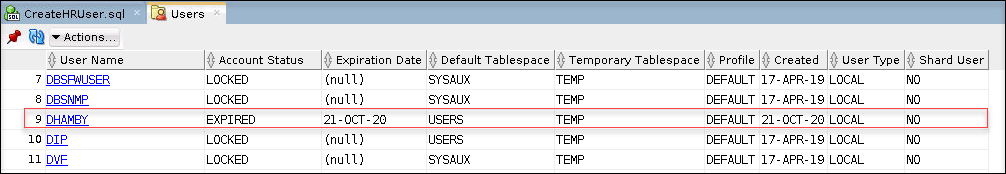
-- SYSTEM PRIVILEGES

GRANT CREATE SESSION TO "&&username" ;

1. Save this script. Click the Save file Icon or **ctrl-S**.
2. 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.

1. Verify that **DHAMBY** is listed in the Users table. The account is unlocked and the password has expired.



1. 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.

1. Source the oraenv script for the orclcdb database.

$ **. oraenv**

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

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

$

1. 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>

1. 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"

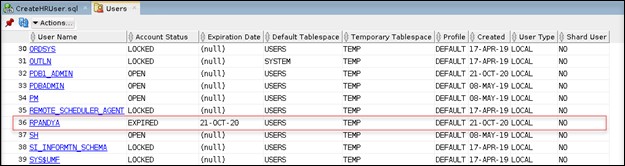
Grant succeeded.

old 1: GRANT CREATE SESSION TO "&&username" new 1: GRANT CREATE SESSION TO "RPANDYA"

Grant succeeded.

SQL>

1. 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.



1. Close the SQL\*Develop 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.

1. Return to the terminal window.
2. 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 dhamby/*original-password*@orclpdb1**

ERROR:

ORA-28001: the password has expired

Changing password for **dhamby**

New password: ***new-password***

Retype new password: ***new-password***

Password changed Connected.

SQL>

1. View the salary for employee 197 from the HR.EMPLOYEES table. The query returns a value of 3000 for SALARY.

SQL> **select salary from hr.employees where employee\_id=197;**

SALARY 3000

SQL>

1. Now attempt to delete the same row from the HR.EMPLOYEES table. DHAMBY is not allowed to perform DELETE operations on this table; therefore, the query returns an "insufficient privileges" error message.

SQL> **delete from hr.employees where employee\_id=197;**

DELETE FROM hr.employees WHERE employee\_id=197

\* ERROR at line 1:

ORA-01031: insufficient privileges

SQL>

1. Disconnect from 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.

1. 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>

1. Select the salary for employee 197 from the HR.EMPLOYEES table. The query returns a value of 3000 for SALARY.

SQL> **select salary from hr.employees where employee\_id=197;**

SALARY 3000

SQL>

1. Delete the same row from the HR.EMPLOYEES table. JGOODMAN has the HRMANAGER role, and that role is granted SELECT, INSERT, UPDATE, and DELETE privileges on all tables in the HR schema. Therefore, the row is deleted.

SQL> **delete from hr.employees where employee\_id=197;**

1 row deleted.

SQL>

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

SQL> **rollback;**

Rollback complete. SQL>

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

SQL> **select count(\*) from hr.employees;**

COUNT(\*)

107

SQL>

1. Disconnect from ORCLPDB1.

SQL> **disconnect**

… SQL>

#### Test the Idle Time Limit in HRPROFILE

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.

1. 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>

1. 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>

1. Wait for 15 minutes. You can leave this terminal window open while waiting.
2. After 15 minutes, query the salary for employee 197 from the HR.EMPLOYEES table. The query returns the message "exceeded maximum idle time..." which indicates that HRPROFILE is working.

SQL> **select salary from hr.employees where employee\_id=197;**

ERROR at line 1:

ORA-02396: exceeded maximum idle time, please connect again SQL>

1. Exit SQL\*Plus

SQL> **exit**

…

$

1. 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>

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

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>

1. Set the default role for JGOODMAN to be HRCLERK only by using the ALTER USER command and DEFAULT ROLE clause.

SQL> **alter user jgoodman default role hrclerk;**

User altered.

SQL>

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

SQL> **select granted\_role, default\_role from dba\_role\_privs where**

**grantee='JGOODMAN';**

GRANTED\_ROLE

DEF

HRCLERK

HRMANAGER

YES

NO

SQL>

1. Disconnect PDBADMIN from ORCLPDB1.

SQL> **disconnect**

… SQL>

#### Enable a Non-Default Role

1. Connect to ORCLPDB1 as JGOODMAN. Refer to *Course Practice Environment: Security Credentials* for the ***password*** value.

SQL> **connect jgoodman/*password*@orclpdb1**

Connected.

SQL>

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

SQL> **column role format a30**

SQL> **select \* from session\_roles;**

ROLE

HRCLERK SQL>

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

SQL> **set role HRMANAGER;**

Role set.

SQL>

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

SQL> **select \* from session\_roles;**

ROLE

HRMANAGER SQL>

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

SQL> **set role HRMANAGER, HRCLERK;**

Role set.

SQL>

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

SQL> **select \* from session\_roles;**

ROLE

HRCLERK HRMANAGER

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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

$

1. Shut down all Oracle processes of all instances.
   1. Shut down the listeners, LISTENER and LISTENER2

[oracle@edvmr1p0 ~]$ **lsnrctl stop listener**

…

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=edvmr1p0.us.oracle.c om)(PORT=1521)))

The command completed successfully [oracle@edvmr1p0 ~]$ **lsnrctl stop listener2**

…

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=edvmr1p0.us.oracle.c om)(PORT=1561)))

The command completed successfully

$

* 1. Shut down the orclcdb instance and exit sqlplus.

$ **sqlplus / as sysdba**

**…**

SQL> **shutdown immediate**

…

SQL> **exit**

$

* 1. Verify that all instances are down.

$ **pgrep -lfa smon**

14610 ora\_smon\_CDBDEV

26743 ora\_smon\_CDBTEST

$

* 1. 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**

…

$

* 1. Confirm all instances are down with another pgreg –lfa smon command.

$ **pgrep –lfa smon**

$

1. Enable the unified auditing feature.

$ **cd $ORACLE\_HOME/rdbms/lib**

$ **make -f ins\_rdbms.mk uniaud\_on ioracle**

…

$

1. Restart the processes.
   1. Change the Oracle Environment to orclcdb.

$ **. oraenv**

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

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

$

* 1. Restart the listener.

$ **lsnrctl start listener**

…

STATUS of the LISTENER

Alias listener

…

$ **lsnrctl start listener2**

…

STATUS of the LISTENER

Alias

…

$

listener2

* 1. Change your working directory back to /home/oracle

$ **cd**

$ **pwd**

/home/oracle

$

* 1. Restart the orclcdb database instance.

$ **sqlplus / as sysdba**

**…**

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

… SQL>

1. 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>

1. Open all PDBs.

SQL> **alter pluggable database all open;**

Pluggable database altered. SQL>

1. Set all databases to open on startup.

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

Pluggable database altered. SQL>

1. Exit from SQL\*Plus.

SQL> **exit**

…

$

1. 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>

1. 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>

1. 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>

1. Exit SQL\*Plus and terminals.

SQL> **exit**

…

$ **exit**

1. 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.
   1. 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

$

* 1. 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>

* 1. 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>

* 1. Enable the audit policy for all users.

SQL> **audit policy jobs\_audit\_upd**;

Audit succeeded. SQL>

1. 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>

1. Test the audit policy by connecting as a user that has privileges to update rows in the

HR.JOBS table.

* 1. 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 JOB\_TITLE MIN\_SALARY

MAX\_SALARY

NOT NULL VARCHAR2(10) NOT NULL VARCHAR2(35)

NUMBER(6)

NUMBER(6)

SQL> **update hr.jobs set max\_salary = 50000 where job\_title**

**='President';**

1 row updated.

SQ>

* 1. 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 ORA\_SECURECONFIG

ORA\_SECURECONFIG JOBS\_AUDIT\_UPD

JGOODMAN SET ROLE 21-OCT-20 09:28 JGOODMAN SET ROLE 21-OCT-20 09:30 JGOODMAN SET ROLE 21-OCT-20 09:30

JGOODMAN SET ROLE 21-OCT-20 10:21

JGOODMAN UPDATE

21-OCT-20 10:33

* 1. 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>

* 1. Run the query in step 3b again to view the audit trail records.

1. Exit from SQL\*Plus.

SQL> **exit**

…

$

1. 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**

…

$

1. 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

$

1. 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**

…

$

1. 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

$

1. 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>

1. 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>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. Change to the /home/oracle/labs/DBMod\_LoadTrans directory & verify.

$ **cd /home/oracle/labs/DBMod\_LoadTrans**

$ **pwd**

/home/oracle/labs/DBMod\_LoadTrans

$

1. 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**.**

$

1. 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>

1. Verify that the table is loaded with the eight records from the products.dat file. The results show that the records were loaded.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PRODUCT\_ID |  | COU |  | LABEL |  | DETAILED\_LABEL |
| 1001 |  | ENG |  | Shutter1 |  | Outdoor shutter1 |
| 1002 |  | FRE |  | Porte1 |  | Porte exterieure1 |
| 1003 |  | SPA |  | Puerta1 |  | Puerta exterior1 |
| 1004 |  | GER |  | Tur1 |  | Auberliche Tur1 |
| 1005 |  | FRE |  | Volet1 |  | Volet exterieur1 |
| 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 |

1. Exit SQL\*Plus.

SQL> **exit**

…

$

SQL> **select \* from products;**

15 rows selected.

SQL>

1. 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:

CPU time was:

00:00:02.10

00:00:00.02

$

1. **Question:** Which operations did SQL\*Loader execute in express mode?

**Answer:** SQL\*Loader first created a temporary external table, used the external table to load the content of the external data file into the table, and finally dropped the temporary external table.

1. In the /home/oracle/labs/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

Terminated by ","

CHAR (255)

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

$

1. 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>

1. Determine the number of rows in the SH.INVENTORIES table. The result shows 476 rows.

SQL> **select count(\*) from inventories;**

COUNT(\*)

476

SQL>

1. Exit from SQL\*Plus.

SQL> **exit**

…

$

1. 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.

$

1. 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>

1. Determine the number of rows in the SH.INVENTORIES table. The result shows 559 rows.

SQL> **select count(\*) from inventories;**

COUNT(\*)

559

SQL>

1. **Question:** Did SQL\*Loader append new rows or replace rows in the SH.INVENTORIES

table?

**Answer:** Originally, there were 476 rows in this table. Now there are 559 rows, which means 83 new rows were added, or "appended," by SQL\*Loader.

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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

$

1. 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)

$

1. 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 quite, 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)

1. 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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

1. 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>

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.

Table SH.INVENTORIES:

83 Rows successfully loaded.

Check the log file: inventories.log

for more information about the load.

$

1. Verify the number of rows in the INVENTORIES table. The table now has 83 rows. The

TRUNCATE option cleared out the original rows in the table and inserted 83 new rows.

SQL> **SELECT count(\*) FROM inventories;**

COUNT(\*)

83

$ **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

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.

1. 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

…

$

1. 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.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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.

$

1. View the inventories.log file. The log file says that 20 rows were successfully loaded into the SH.INVENTORIES table; however, 51 rows were not loaded due to errors.

$ **cat inventories.log**

SQL\*Loader: Release 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:

CPU time was:

$

00:00:00.39

00:00:00.03

1. **Question:** Did SQL\*Loader try to load all rows?

**Answer:** No. After 20 rows successfully loaded, 51 rows did not load due to a constraint violation error. The load stopped at this point. The default number of errors tolerated is 50. When the number was exceeded, SQL\*Loader stopped.

#### Load Data by Using SQL\*Loader in Direct Mode

Observe how SQL\*Loader behaves when loading the SH.INVENTORIES table in direct mode.

1. Start SQL\*Loader, connect to 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.

Load completed - logical record count 83.

Table SH.INVENTORIES:

83 Rows successfully loaded.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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. |

Check the log file: inventories.log

for more information about the load.

$

1. **Question:** Does the direct load use the SQL INSERT statement? How does the direct path commit the rows inserted?

**Answer:** The direct load loads records into the blocks, writing the data blocks directly to the database files. You can observe that there is no COMMIT instruction, but SAVE instead.

During a data save, only full database blocks are written to the database.

1. **Question:** Did it enforce the CHECK constraint?

**Answer:** No, it did not. This is the reason all rows were loaded, regardless of the

WAREHOUSE\_ID value to be inserted.

1. **Question:** Does SQL\*Loader in direct mode ignore all constraints?

**Answer:** No, it does not. It enforces PRIMARY KEY, UNIQUE, and NOT NULL constraints.

1. 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

$

1. 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

$

1. 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

$

1. **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.

1. 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>

1. 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>

1. Grant the OE user READ WRITE privileges on the DP\_FOR\_OE directory.

SQL> **grant read, write on directory dp\_for\_oe to oe;**

Grant succeeded.

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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"

105 rows

. . exported "OE"."ORDER\_ITEMS" 665 rows

Master table "OE"."SYS\_EXPORT\_SCHEMA\_01" successfully loaded/unloaded

12.73 KB

21.01 KB

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

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

Dump file set for OE.SYS\_EXPORT\_SCHEMA\_01 is:

/u01/app/oracle/admin/orclcdb/dpdump/expoe.dmp

Job "OE"."SYS\_EXPORT\_SCHEMA\_01" successfully completed at Thu Oct 22 00:59:21 2020 elapsed 0 00:01:07

1. **Question:** How can you verify that objects other than tables, such as constraints, indexes, and sequences, were exported?

**Answer:** Generate a SQL script from the dump file by performing an import and specifying the SQLFILE parameter.

1. Use Data Pump Import to generate a SQL script named oe\_SQL.sql from the dump file. Refer to “Course Practice Environment: Security Credentials” for the ***password*** value.

$ **impdp oe/*password*@orclpdb1 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

1. 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('US ERENV','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 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS

2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT) TABLESPACE "TBS\_APP2" ;

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 INITRANS 1 MAXTRANS 255

NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS

2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT) TABLESPACE "TBS\_APP" ;

-- new object type path: SCHEMA\_EXPORT/TABLE/INDEX/INDEX

CREATE INDEX "OE"."I\_ORDER\_ITEMS" ON "OE"."ORDER\_ITEMS" ("ORDER\_ID")

PCTFREE 10 INITRANS 2 MAXTRANS 255

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS

2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT) TABLESPACE "TBS\_APP" 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 INITRANS 2 MAXTRANS 255

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS

2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT) TABLESPACE "TBS\_APP" ENABLE;

-- new object type path: SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

-- new object type path: SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

-- new object type path: SCHEMA\_EXPORT/STATISTICS/MARKER

-- fixup virtual columns... done fixup virtual columns

$

#### Import the OE Schema into ORCLPDB2 by Using Data Pump Import

1. 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>

1. 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>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. Use Data Pump to import the OE schema. Refer to “Course Practice Environment: Security Credentials” for the ***password*** value. Use the REMAP\_SCHEMA parameter to import the entire OE schema into a new OETEST schema in 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

$

1. **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.

1. Connect to ORCLPDB2 as the SYSTEM user. Refer to “Course Practice Environment: Security Credentials” for the ***password*** value.

$ **sqlplus system/*password*@orclpdb2**

… SQL>

1. Create the DP\_FOR\_OE directory in ORCLPDB2.

SQL> **create directory dp\_for\_oe AS '/u01/app/oracle/admin/orclcdb/dpdump';**

Directory created. SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS

2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT) TABLESPACE "TBS\_APP2"

Processing object type SCHEMA\_EXPORT/TABLE/TABLE\_DATA

. . imported "OETEST"."ORDERS" 12.73 KB

105 rows

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX

ORA-39112: Dependent object type INDEX:"OETEST"."I\_ORDER\_ITEMS" skipped, base object type TABLE:"OETEST"."ORDER\_ITEMS" creation failed

Processing object type SCHEMA\_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/STATISTICS/INDEX\_STATISTICS

Processing object type SCHEMA\_EXPORT/TABLE/STATISTICS/TABLE\_STATISTICS

Processing object type SCHEMA\_EXPORT/STATISTICS/MARKER

Job "SYSTEM"."SYS\_IMPORT\_FULL\_01" completed with 2 error(s) at Thu Oct 22 01:22:12 2020 elapsed 0 00:00:27

$

1. **Question:** Did the import complete successfully?

**Answer:** Not completely. Data Pump imported only the objects that it could process without any error.

1. **Question:** Which objects were not imported?

**Answer:** Data Pump could not import the ORDER\_ITEMS table because this table requires the TBS\_APP2 tablespace, which does not exist in ORCLPDB2. The dependent objects of this table, such as an index could not be imported.

1. Create the TBS\_APP2 tablespace in ORCLPDB2.
   1. 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>

* 1. 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>

* 1. 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

$

1. **Question:** Are the errors true errors?

**Answer:** The errors are normal errors stating that objects exist. They were created during the previous import operation.

#### Verify the OETEST Schema in ORCLPDB2

Verify that the new OETEST schema exists in ORCLPDB2.

1. 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**

…

$

1. View the list of tables to which the OETEST user has access.

SQL> **column table\_name format a15**

SQL> **select table\_name from user\_tables**; TABLE\_NAME

ORDERS ORDER\_ITEMS

SQL>

1. Query the number of rows in the ORDER\_ITEMS table. The results show that there are 665 rows.

SQL> **select count(\*) from order\_items;**

COUNT(\*) 665

SQL>

1. List the indexes to which the OETEST user has access.

SQL> **column index\_name format a20**

SQL> **SELECT index\_name FROM user\_indexes;**

INDEX\_NAME

SYS\_C007956 I\_ORDER\_ITEMS

SQL>

1. List the sequences to which the OETEST user has access.

SQL> **column sequence\_name format a20**

SQL> **select sequence\_name from user\_sequences;**

SEQUENCE\_NAME

ORDERS\_SEQ SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. **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

1. 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>

1. 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>

1. Drop the target user created in the previous import operation.

SQL> **drop user oetest cascade;**

User dropped.

SQL>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. Invoke Data Pump Import and use the NETWORK\_LINK parameter to initiate an import via a database link. Refer to “Course Practice Environment: Security Credentials” for the ***password*** value.

$ **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"

rows

. . imported "OETEST"."ORDER\_ITEMS"

rows

Processing object type SCHEMA\_EXPORT/TABLE/INDEX/INDEX

105

665

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

$

1. Verify that the OE schema was imported as OETEST into ORCLPDB2.
   1. 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>

* 1. 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>

* 1. Query the number of rows in the ORDER\_ITEMS table. The table has 665 rows.

SQL> **select count(\*) from order\_items;**

COUNT(\*)

665

SQL>

* 1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. **Question:** What are the advantages and drawbacks of this type of Data Pump import?

**Answer:** There are no dump files involved. If an import operation is performed over an unencrypted network link, then all data is imported as clear text even if it is encrypted in the database.

## Practice 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.

$

1. Start a SQL\*Plus session and verify the prerequisites for transporting a tablespace across platforms.
2. 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  SQL> |  | NOARCHIVELOG |  | READ WRITE |  | 4116221 |

1. 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

1. Linux IA (32-bit
2. Linux IA (64-bit)

9 IBM zSeries Based Linux

13 Linux x86 64-bit

18 IBM Power Based Linux

22 Linux OS (S64)

Little Little Big Little Big

Big

6 rows selected.

SQL>

1. 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**

…

$

1. 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>

1. 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>

1. 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*.

1. 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>

1. 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**

…

$

1. 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\_htlB" successfully loaded/unloaded

IMPDP> Starting "SYS"."TSPITR\_IMP\_ORCLCDB\_htlB":

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\_htlB" successfully completed at Thu Oct 22 02:14:30 2020 elapsed 0 00:00:23

Import completed

Finished restore at 22-OCT-20 RMAN>

1. 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 SYSAUX UNDOTBS1 TEMP USERS TBS\_APP TBS\_APP2

BARTBS

ONLINE ONLINE ONLINE ONLINE ONLINE ONLINE ONLINE

READ ONLY

8 rows selected

RMAN> **exit**

Recovery Manager complete.

$

1. 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.

$

1. 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

$

1. 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.

1. 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>

1. In ORCLPDB1, create the SH.SALES\_EXT\_RANGE external table.
   1. 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>

* 1. 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>

* 1. 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>

* 1. 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>

* 1. 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>

* 1. **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.

* 1. 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\_xternal\_loc\_partitions;**

TABLE\_NAME

PARTITION\_NAME LOCATION

DIRECTORY\_NAME

SALES\_EXT\_RANGE YEAR1998

SALES\_EXT\_RANGE YEAR1999

DP\_sales\_1998.dat

DP2\_sales\_1999.dat EXT\_DIR2

SQL>

1. Determine the number of rows in the external table based on specific criteria.
   1. Determine the number of rows for sales in 1998.

SQL> **select count(\*) from sh.sales\_ext\_range partition (year1998);**

COUNT(\*) 357668

SQL>

* 1. Determine the number of rows for sales in 1999.

SQL> **select count(\*) from sh.sales\_ext\_range partition (year1999);**

COUNT(\*) 495890

SQL>

* 1. Determine the number of rows for sales in both 1998 and 1999.

SQL> **SELECT count(\*) FROM sh.sales\_ext\_range;**

COUNT(\*) 853558

SQL>

* 1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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 853574. 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

1. Check the log files to determine the reason for the discrepancy.
   1. 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

* 1. 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)

...

$

1. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. See Product-Specific Credentials for the ***password***.

$ **sqlplus system/*password*@orclpdb1**

... SQL>

1. 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>

1. 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>

1. 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>

1. 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>

1. 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>

1. Exit SQL\*Plus.

SQL> **exit**

…

$

1. 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

…

1. 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

$

1. Start SQL\*Plus and connect to ORCLPDB1 as the SYSTEM user. See Product-Specific Credentials for the ***password***.

$ **sqlplus system/*password*@orclpdb1**

... SQL>

1. 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>

1. 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>

1. Determine the number of rows in the OE.ORDERS\_EXT table

SQL> **select count(\*) from oe.orders\_ext;**

COUNT(\*)

105

SQL>

1. Connect to ORCLPDB2 as the SYSTEM user. See Product-Specific Credentials for the

***password***.

SQL> **CONNECT SYSTEM/*password*@orclpdb2**

Connected.

SQL>

1. 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>

1. Create an external directory named ext\_dir.

SQL> **create directory ext\_dir as '/home/oracle/labs/DBMod\_LoadTrans/DP/';**

Directory created. SQL>

1. 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>

1. 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>

1. **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.

1. Re-create the external table with the appropriate access driver.
   1. Drop the OE.ORDERS\_EXT table that you just created.

SQL> **drop table oe.orders\_ext;**

Table dropped.

SQL>

* 1. 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>

1. 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

105 rows selected.

SQL>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 5458 16-AUG-07 |  | 02.34.12.234359 | PM | direct | 101 |
| 0 | 78280 |  | 153 |  |  |  |
|  | 5397 19-NOV-07 |  | 03.41.54.696211 | PM | direct | 102 |
| 1 | 42283 |  | 154 |  |  |  |
|  | 5454 02-OCT-07 |  | 04.49.34.678340 | PM | direct | 103 |
| 1  … | 6653  5456 07-NOV-06 |  | 154  08.53.25.989889 | PM | direct | 117 |
| 0 | 3878 |  | 163 |  |  |  |
|  | 5457 31-OCT-07 |  | 10.22.16.162632 | PM | direct | 118 |
| 5 | 21586 |  | 159 |  |  |  |

1. 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

$

1. 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

auto optimizer stats collection auto space advisor

ENABLED

ENABLED ENABLED

SQL>

1. 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>

1. 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

auto optimizer stats collection auto space advisor

DISABLED

ENABLED ENABLED

SQL>

1. 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>

1. 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

auto optimizer stats collection auto space advisor

ENABLED

ENABLED ENABLED

SQL>

1. 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

$

1. 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>

1. 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> |  |  |  |  |  |  |  |  |

1. Increase the Sunday maintenance window by 2 hours.
   1. 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>

* 1. 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>

* 1. 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>

1. 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

There are no practices for Lesson 30.

# 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

$

1. Start SQL\*Plus and connect to ORCLCDB database as the SYS user with SYSDBA privileges .

$ **sqlplus / as sysdba**

… SQL>

1. 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**

…

$

1. 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\_dia0\_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 ? (DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROT | | | | | | 00:00:00 oracleorclcdb OCOL=beq))) | |
| oracle | 22933 | 1 | 0 | Oct29 | ? | 00:00:00 | ora\_w006\_orclcdb |
| oracle | 23036 | 1 | 0 | Oct29 | ? | 00:00:00 | ora\_w007\_orclcdb |
| oracle orclcdb  $ | 30569 | 21226 | 0 | 01:09 | pts/2 | 00:00:00 | grep --color=auto |

Notice the output included the process id of the grep command.

1. 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 -lf 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

$

1. 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

$

1. 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.

* 1. 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).

* 1. 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

* 1. Determine if there are archiver processes arcn (where n is a number 0 through 9).

$ pgrep -lf arc[0-9]\_orclcdb

$

* 1. 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 Automatic archival Archive destination

Oldest online log sequence Current log sequence

SQL>

No Archive Mode Disabled

USE\_DB\_RECOVERY\_FILE\_DEST 29

31

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 |

1. All the running background processes can be viewed using the v$bgprocess view.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| …  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  …  CKPT |  | 0  1 |  | Atomic Controlfile to Memory  checkpoint | Server |  |  |
| … 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  … LG00 |  | 0  1 |  | db writer process 4  Log Writer Slave |  |  | SLAVE |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LG01 | 1 | Log Writer Slave |  | SLAVE |
| LGWR  … LREG | 1  1 | Redo etc.  Listener Registration |  |  |
| … MMAN | 1 | Memory Manager |  |  |
| MMNL | 2 | Manageability Monitor | Process 2 |  |
| MMON  … PMAN | 1  1 | Manageability Monitor  process manager | Process |  |
| PMON  …  RECO | 1  1 | process cleanup  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.

1. Exit SQL\*plus

SQL> **exit**

…

$

1. 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

$

1. 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>

1. Examine the v$session view

SQL> **desc v$session**

Name

Null?

Type

SADDR SID SERIAL# AUDSID PADDR USER# USERNAME COMMAND OWNERID TADDR LOCKWAIT STATUS SERVER

SCHEMA#

RAW(8) NUMBER NUMBER NUMBER RAW(8) NUMBER

VARCHAR2(128) NUMBER NUMBER VARCHAR2(16)

VARCHAR2(16) VARCHAR2(8) VARCHAR2(9)

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 SHARD\_DDL\_STATUS CON\_ID EXTERNAL\_NAME

PLSQL\_DEBUGGER\_CONNECTED SQL>

NUMBER VARCHAR2(8) NUMBER VARCHAR2(1024)

VARCHAR2(5)

1. 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>

1. 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 PORT

OSUSER

MACHINE

PROCESS

PROGRAM

SYS oracle

(TNS V1-V3)

edvmr1p0 51468

24117

sqlplus@edvmr1p0

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.

1. Use the ps command to display the information about the OS Process Id for your session.

SQL> **host**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| $ **ps –ef** | **| grep 24117** | | | | | | |
| oracle sysdba | 24117 | 11237 | 0 | 03:31 | pts/3 | 00:00:00 | sqlplus as |
| oracle | 28395 | 24117 | 0 | 04:05 | pts/3 | 00:00:00 | /bin/bash |
| oracle 24117 | 28844 | 28395 | 0 | 04:05 | pts/3 | 00:00:00 | grep --color=auto |
| $ 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.

1. Exit SQL\*plus.

SQL> **exit**

…

$

1. 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>

1. 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 memory\_target sga\_max\_size

sga\_target

0

0

2013265920

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.

1. 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 large\_pool\_size java\_pool\_size streams\_pool\_size

db\_cache\_size

0

0

0

0

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.

1. Examine the view v$memory\_dynamic\_components

SQL> **desc v$memory\_dynamic\_components**

SQL>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

1. Determine how much memory is assigned to each pool/cache in the instance by querying the view v$memory\_dynamic\_components

#### SQL> column component format a30

SQL> **select component, current\_size, min\_size, max\_size, last\_oper\_type from v$memory\_dynamic\_components order by 1 desc;**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| COMPONENT LAST\_OPER\_TYP |  | CURRENT\_SIZE |  | MIN\_SIZE |  | MAX\_SIZE |
|  |  |  |  |  |  |  |
| unified pga pool STATIC |  | 0 |  | 0 |  | 0 |
| streams pool GROW |  | 33554432 |  | 16777216 |  | 33554432 |
| shared pool GROW |  | 687865856 |  | 637534208 |  | 687865856 |
| memoptimize buffer cache STATIC |  | 0 |  | 0 |  | 0 |
| large pool SHRINK |  | 16777216 |  | 16777216 |  | 33554432 |
| java pool STATIC |  | 0 |  | 0 |  | 0 |
| Shared IO Pool STATIC |  | 100663296 |  | 100663296 |  | 100663296 |
| SGA Target STATIC |  | 2013265920 |  | 2013265920 |  | 2013265920 |
| RECYCLE buffer cache STATIC |  | 0 |  | 0 |  | 0 |
| PGA Target STATIC |  | 671088640 |  | 671088640 |  | 671088640 |
| KEEP buffer cache STATIC |  | 0 |  | 0 |  | 0 |
| In-Memory Area STATIC |  | 0 |  | 0 |  | 0 |
| In Memory RW Extension Area STATIC |  | 0 |  | 0 |  | 0 |
| In Memory RO Extension Area STATIC |  | 0 |  | 0 |  | 0 |
| Data Transfer Cache STATIC |  | 0 |  | 0 |  | 0 |
| DEFAULT buffer cache GROW |  | 1157627904 |  | 1140850688 |  | 1224736768 |
| DEFAULT 8K buffer cache STATIC |  | 0 |  | 0 |  | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DEFAULT STATIC | 4K buffer cache | 0 | 0 | 0 |
| DEFAULT STATIC | 32K buffer cache | 0 | 0 | 0 |
| DEFAULT STATIC | 2K buffer cache | 0 | 0 | 0 |
| DEFAULT STATIC | 16K buffer cache | 0 | 0 | 0 |
| ASM Buffer Cache  STATIC | | 0 | 0 | 0 |

1. 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>

22 rows selected.

SQL>

1. 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.

1. 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 END\_TIME

CON\_ID

DATE DATE

NUMBER

SQL>

1. 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;**

13 rows selected.

SQL>

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | |

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.

1. Exit SQL\*plus.

SQL> **exit**

…

$

1. Close all terminals.

# Practices for Lesson 33: Analyzing SQL and Optimizing Access Paths

## Practices for Lesson 33: Overview

### Overview

In these practices, you will learn how to optimize SQL performance.

## Practice 33-1: 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, the 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.

$

1. 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.

1. Open another new terminal window. This window will be referred to as Window 2.
2. 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>

1. 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>

1. 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\_implementation\_result CLOB; BEGIN

v\_tname := DBMS\_STATS.CREATE\_ADVISOR\_TASK(v\_tname); sh\_obj\_filter(v\_tname);

v\_ename := DBMS\_STATS.EXECUTE\_ADVISOR\_TASK(v\_tname); END;

/

SQL> **@$HOME/labs/DBMod\_MonTune/OPTADV\_2.sql**

PL/SQL procedure successfully completed. SQL>

1. Verify that the procedure completed successfully.
   1. 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>

* 1. 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\_NA EXECUTION TYPE

STATUS

…

SYS\_AUTO\_SPM\_EVOLVE\_TASK EXEC\_143

21-OCT-20 SPM EVOLVE COMPLETED

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AUTO\_STATS\_ADVISOR\_TASK SYS\_AUTO\_SPM\_EVOLVE\_TASK AUTO\_STATS\_ADVISOR\_TASK MY\_TASK | EXEC\_144 EXEC\_153 EXEC\_154 EXEC\_165 | 1. OCT-20 2. OCT-20 3. OCT-20 4. OCT-20 | STATISTICS SPM EVOLVE STATISTICS STATISTICS | COMPLETED COMPLETED COMPLETED COMPLETED |
| 29 rows selected.  SQL> |  |  |  |  |
| 8. Generate a report. |  |  |  |  |

1. 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>

1. 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>

1. 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**';

1. 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.
4. There are 1 statistics operation(s) using nondefault parameters. Use default parameters for statistics operations.
5. There are 1 uses of GATHER\_TABLE\_STATS. Use GATHER\_SCHEMA\_STATS instead of GATHER\_TABLE\_STATS.
6. There are 10 object(s) with no statistics. Gather Statistics on those objects with no statistics.

6 rows selected.

1. Generate the script before a possible implementation.
   1. 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>

* 1. 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 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 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 AVOIDSTALESTATS

-- Rule Description: Avoid objects with stale or no statistics

-- Gather statistics for those objcts that are missing or have no statistics.

-- Scripts for rule MAINTAINSTATSCONSISTENCY

-- Rule Description: Statistics of dependent objects should be consistent

-- Gather statistics for those objcts that are missing or have no statistics.

declare obj\_filter\_list

dbms\_stats.ObjectTab;

obj\_filter dbms\_stats.ObjectElem; obj\_cnt number := 0;

begin

obj\_filter\_list := dbms\_stats.ObjectTab();

obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'CUSTOMER';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE';

obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'CUSTOMERS'; obj\_filter\_list.extend();

obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE'; obj\_filter.objname :=

'DATE\_DIM';

obj\_filter\_list.extend();

obj\_cnt := obj\_cnt + 1; obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname :=

'OE';

obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'INVENTORIES'; obj\_filter\_list.extend();

obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE'; obj\_filter.objname :=

'LINEORDER';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname :=

'OE';

obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'ORDERS'; obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE'; obj\_filter.objname :=

'ORDER\_ITEMS';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname

:= 'OE';

obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'PART'; obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter; obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE';

obj\_filter.objname := 'PRODUCT\_INFORMATION';

obj\_filter\_list.extend(); obj\_cnt := obj\_cnt + 1;

obj\_filter\_list(obj\_cnt) := obj\_filter;

obj\_filter.ownname := 'OE'; obj\_filter.objtype := 'TABLE'; obj\_filter.objname := 'SUPPLIER'; obj\_filter\_list.extend();

obj\_cnt := obj\_cnt + 1; obj\_filter\_list(obj\_cnt) := obj\_filter; dbms\_stats.gather\_database\_stats(

obj\_filter\_list=>obj\_filter\_list); end;

/

PL/SQL procedure successfully completed.

1. Question: What does obj\_filter\_list indicate?

Answer: The object filter list contains the names of the ten objects with no statistics.

1. Question: What would you do if you agree with the recommendations? Answer: You could either execute the generated SQL script or use the

DBMS\_STATS.IMPLEMENT\_ADVISOR\_TASK procedure.

1. Execute the $HOME/labs/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.

1. Check that the statistics are collected for the ten objects. Columns such as NUM\_ROWS, EMPTY\_BLOCKS, BLOCKS, and AVG\_ROW\_LEN have null values until the statistics are collected.

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. |  |  |  |  |  |  |  |  |  |  |

1. Drop the advisor task.

SQL> exec DBMS\_STATS.DROP\_ADVISOR\_TASK('MY\_TASK')

PL/SQL procedure successfully completed. SQL>

1. Exit SQL\*Plus and close Window 2.

SQL> **exit**

…

$ **exit**

#### Window 1: Stop the Workload

1. In Window 1, press Ctrl+c to stop the activity and then close the window.