



Oracle Database 23ai: Administration Workshop

Activity Guide

S1106066GC10

Learn more from Oracle University at education.oracle.com



This document is for instructor use only.
It is not intended for general distribution.

Copyright © 2024, Oracle and/or its affiliates.

Disclaimer

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

The information contained in this document is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

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

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Trademark Notice

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

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

1012202024

For Instructor Use Only.
This document should not be distributed.

Table of Contents

Course Practice Environment: Security Credentials	7
Course Practice Environment: Security Credentials.....	8
Practices for Lesson 1: Introduction to Oracle Database	9
Overview	10
Practices for Lesson 2: Accessing an Oracle Database	11
Overview	12
Practices for Lesson 3: Creating an Oracle Database by Using DBCA	13
Overview	14
Practice 3-1: Creating a New CDB by Using DBCA.....	15
Practices for Lesson 4: Creating an Oracle Database by Using an SQL Command	21
Overview	22
Practice 4-1: Creating a New CDB by Using an SQL Command.....	23
Practices for Lesson 5: Starting Up and Shutting Down an Oracle Database	29
Overview	30
Practice 5-1: Shutting Down and Starting Up the Oracle Database	31
Practices for Lesson 6: Managing Database Instances	41
Overview	42
Practice 6-1: Investigating Initialization Parameter Files	43
Practice 6-2: Viewing Initialization Parameters by Using SQL*Plus.....	51
Practice 6-3: Modifying Initialization Parameters by Using SQL*Plus	68
Practice 6-4: Viewing Diagnostic Information	79
Practices for Lesson 7: Oracle Net Services Overview	89
Overview	90
Practices for Lesson 8: Configuring Naming Methods	91
Overview	92
Practice 8-1: Configuring the Oracle Network to Access a Database	93
Practice 8-2: Creating a Net Service Name for a PDB	95
Practices for Lesson 9: Configuring and Administering the Listener	99
Overview	100
Practice 9-1: Exploring the Default Listener	101
Practice 9-2: Creating a Second Listener	109
Practices for Lesson 10: Configuring a Shared Server Architecture.....	119
Overview	120
Practice 10-1: Configuring Shared Server Mode	121

Practice 10-2: Configuring Clients to Use a Shared Server	124
Practices for Lesson 11: Creating PDBs from Seed	127
Overview	128
Practice 11-1: Creating a New PDB from the PDB Seed	129
Practices for Lesson 12: Using Other Techniques to Create PDBs	133
Overview	134
Practices for Lesson 13: Managing PDBs	135
Overview	136
Practice 13-1: Renaming a PDB	137
Practice 13-2: Setting Parameter Values for PDBs	140
Practices for Lesson 14: Database Storage Overview	147
Overview	148
Practices for Lesson 15: Creating and Managing Tablespaces	149
Overview	150
Practice 15-1: Viewing Tablespace Information	151
Practice 15-2: Creating a Tablespace	160
Practice 15-3: Managing Temporary and Permanent Tablespaces	170
Practices for Lesson 16: Improving Space Usage	179
Overview	180
Practice 16-1: Managing Space in Tablespaces	181
Practice 16-2: Using Compression	194
Practice 16-3: Enabling the Resumable Space Allocation Feature	202
Practices for Lesson 17: Managing Undo Data	209
Overview	210
Practice 17-1: Viewing Undo Tablespaces in a CDB	211
Practices for Lesson 18: Creating and Managing User Accounts	213
Overview	214
Practice 18-1: Creating Common and Local Users	215
Practice 18-2: Creating a Local User for an Application	225
Practice 18-3: Exploring OS and Password File Authentication.....	235
Practices for Lesson 19: Configuring Privilege and Role Authorization	239
Overview	240
Practice 19-1: Granting a Local Role (DBA) to PDBADMIN	241
Practice 19-2: Using SQL*Developer to Create Local Roles.....	244
Practices for Lesson 20: Configuring User Resource Limits and Default Role	253
Overview	254
Practice 20-1: Using SQL*Developer to Create a Local Profile.....	255
Practice 20-2: Configuring a Default Role for a User	265

Practices for Lesson 21: Implementing Oracle Database Auditing	269
Overview	270
Practice 21-1: Enabling Unified Auditing	271
Practice 21-2: Creating Audit Users	275
Practice 21-3: Creating an Audit Policy	277
Practices for Lesson 22: Introduction to Loading and Transporting Data.....	283
Overview	284
Practices for Lesson 23: Loading Data	285
Overview	286
Practice 23-1: Loading Data into a PDB from an External File.....	287
Practices for Lesson 24: Transporting Data.....	303
Overview	304
Practice 24-1: Moving Data from One PDB to Another PDB.....	305
Practice 24-2: Transporting a Tablespace	320
Practices for Lesson 25: Using External Tables to Load and Transport Data.....	327
Overview	328
Practice 25-1: Querying External Tables	329
Practice 25-2: Unloading External Tables	337
Practices for Lesson 26: Automated Maintenance Tasks Overview	343
Overview	344
Practices for Lesson 27: Managing Tasks and Windows	345
Overview	346
Practice 27-1: Enabling and Disabling Automated Maintenance Tasks	347
Practice 27-2: Modifying the Duration of a Maintenance Window	349
Practices for Lesson 28: Database Monitoring and Performance Tuning Overview	353
Overview	354
Practices for Lesson 29: Monitoring Database Performance	355
Overview	356
Practices for Lesson 30: Analyzing SQL and Optimizing Access Paths.....	357
Overview	358

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

**Course Practice
Environment: Security
Credentials**

Course Practice Environment: Security Credentials

For operating system (OS) usernames and passwords:

- If you are attending a classroom-based or live virtual class (LVC), ask your instructor or LVC producer for OS credential information.
- If you are using a self-study format, refer to the communication that you received from Oracle University for this course.

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

Product-Specific Credentials		
Product/Application	Username	Password
Database: All databases	ALL DATABASE USERS (COMMON AND LOCAL)	WELCOME123##
Database: Practice 3-1, Step 3, all passwords in DBCA command	PDBADMIN	WELCOME123##

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 1: Introduction to Oracle Database

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 2: Accessing an Oracle Database

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

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

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 3-1: Creating a New CDB by Using DBCA

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 `orclcldb`. 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/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 CDB is created with no PDBs, except the PDB seed.

Tasks

1. Open a terminal window as the `oracle` OS user and execute the `$HOME/labs/DBMod_CreateDB/glogin.sh` shell script to set the formatting of the output.

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_CreateDB/glogin.sh
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 ~]$
```

2. Create the CDB by using DBCA in silent mode.
 - a. Change the directory to `/home/oracle/labs/DBMod_CreateDB`.

```
[oracle@edvmr1p0 ~]$ cd /home/oracle/labs/DBMod_CreateDB
[oracle@edvmr1p0 DBMod_CreateDB]$
```

- b. View content of the /home/oracle/labs/DBMod_CreateDB/CrCDBTEST.sh script, which will be used to create the CDBTEST CDB database.

```
[oracle@edvmr1p0 DBMod_CreateDB]$ cat CrCDBTEST.sh
$ORACLE_HOME/bin/dbca -silent -createDatabase -
templateName General_Purpose.dbc -gdbname CDBTEST -sid
CDBTEST -createAsContainerDatabase true -numberOfPDBs 0 -
useLocalUndoForPDBs true -responseFile NO_VALUE -totalMemory
1800 -sysPassword WELCOME123## -systemPassword WELCOME123##
-pdbAdminPassword WELCOME123## -enableArchive true -
recoveryAreaDestination /u01/app/oracle/fast_recovery_area -
recoveryAreaSize 15000 -datafileDestination
/u01/app/oracle/oradata

$
```

- c. Execute the script.

The script may take approximately 10–15 minutes to complete.

```
[oracle@edvmr1p0 DBMod_CreateDB]$ ./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
69% 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
```

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

```
[oracle@edvmr1p0 DBMod_CreateDB]$ cat /etc/oratab
...
# Multiple entries with the same $ORACLE_SID are not allowed.
#
#
orclcdb:/u01/app/oracle/product/23.4.0/dbhome_1:Y
CDBTEST:/u01/app/oracle/product/23.4.0/dbhome_1:N
[oracle@edvmr1p0 DBMod_CreateDB]$
```

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

- a. Verify that the database is a CDB.

Note: The CDB name CDBTEST was created in caps.

```
[oracle@edvmr1p0 DBMod_CreateDB]$ . 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 23.0.0.0.0 - Limited Availability on Mon Feb
12 23:34:48 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, Oracle. All rights reserved.

Connected to:
Oracle Database 23c Enterprise Edition Release 23.0.0.0.0 -
Limited Availability
Version 23.4.0.23.11

SQL> SELECT cdb FROM v$database;

CDB
---
YES

SQL>
```

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

```
SQL> col name format a58
SQL> SELECT name FROM v$logfile ORDER BY 1;

NAME
-----
/u01/app/oracle/oradata/CDBTEST/pdbseed/sysaux01.dbf
/u01/app/oracle/oradata/CDBTEST/pdbseed/system01.dbf
/u01/app/oracle/oradata/CDBTEST/pdbseed/undotbs01.dbf
/u01/app/oracle/oradata/CDBTEST/sysaux01.dbf
/u01/app/oracle/oradata/CDBTEST/system01.dbf
/u01/app/oracle/oradata/CDBTEST/undotbs01.dbf
/u01/app/oracle/oradata/CDBTEST/users01.dbf

7 rows selected.

SQL>
```

- c. Verify that the tablespaces are created.

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

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

SQL>
```

5. Exit SQL*Plus.

```
SQL> exit
[oracle@edvmr1p0 DBMod_CreateDB]$
```

6. Run the `dropCDBTEST.sh` script to clean up your environment.

```
[oracle@edvmr1p0 DBMod_CreateDB]$ ./dropCDBTEST.sh  
[oracle@edvmr1p0 DBMod_CreateDB]$
```

7. Close the terminal window.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

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

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 4-1: Creating a New CDB by Using an SQL Command

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 `orclcldb`. 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 CDB is created with no PDBs, except the PDB seed.

Tasks

1. Create the CDB by using the `CREATE DATABASE` command.
 - a. Open a terminal window as the `oracle` OS user. Set the environment variables using the `oraenv` script.

```
[oracle@edvmr1p0 ~]$ . oraenv
ORACLE_SID = [CDBTEST] ? CDBDEV
ORACLE_HOME = [/home/oracle] ? /u01/app/oracle/product/23.4.0/dbhome_1
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 ~]$
```

- b. Change the directory to the `$ORACLE_HOME/dbs` directory and create an initialization parameter file named `initCDBDEV.ora` from the sample `init.ora` file.

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/dbs
[oracle@edvmr1p0 dbs]$ cp init.ora initCDBDEV.ora
[oracle@edvmr1p0 dbs]$
```

- c. Use the editor of your choice (`vi` and `gedit` are available) and add the following initialization parameters to the end of the `$ORACLE_HOME/dbs/initCDBDEV.ora` file.

```
[oracle@edvmr1p0 dbs]$ vi initCDBDEV.ora
...
DB_CREATE_FILE_DEST='/u01/app/oracle/oradata'
ENABLE_PLUGGABLE_DATABASE=true
```

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

```
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='23.0.0.0'
```

Note: Change <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.

- e. Save the modified file.

- f. 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.

Note: Your output of the `ls` command may be slightly different.

```
[oracle@edvmr1p0 dbs]$ mkdir -p
/u01/app/oracle/admin/CDBDEV/adump
[oracle@edvmr1p0 dbs]$ ls /u01/app/oracle/fast_recovery_area
ORCLCDB
[oracle@edvmr1p0 dbs]$ ls /u01/app/oracle/oradata
ORCLCDB
[oracle@edvmr1p0 dbs]$
```

- g. Start the database instance in NOMOUNT mode.

```
[oracle@edvmr1p0 dbs]$ sqlplus / AS SYSDBA
...
Connected to an idle instance.

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

- 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>
```

Note: 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.

- Execute catalog and catproc scripts. The catalog.sql script takes ~ 3 minutes to complete while the catproc.sql script takes ~ 30 minutes to complete.

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

- Exit from SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 dbs]$
```

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

```
[oracle@edvmr1p0 dbs]$ echo
"CDBDEV:/u01/app/oracle/product/23.4.0/dbhome_1:N" >> /etc/oratab
[oracle@edvmr1p0 dbs]$
```

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

```
[oracle@edvmr1p0 dbs]$ cat /etc/oratab
...
#
orclcd: /u01/app/oracle/product/23.4.0/dbhome_1:Y
CDBDEV: /u01/app/oracle/product/23.4.0/dbhome_1:N

[oracle@edvmr1p0 dbs]$
```

4. Verify that the characteristics of the database are correct.
 - a. Set the environment variables for your new database.

```
[oracle@edvmr1p0 dbs]$ . oraenv  
ORACLE_SID = [CDBDEV] ? CDBDEV  
The Oracle base remains unchanged with value /u01/app/oracle  
[oracle@edvmr1p0 dbs]$
```

- b. Verify that the database is a CDB.

```
[oracle@edvmr1p0 dbs]$ sqlplus / as sysdba  
...  
Version 23.4.0.23.11  
  
SQL> select cdb from v$database;  
  
CDB  
---  
YES  
  
SQL>
```

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

```
SQL> set pagesize 100  
SQL> column name format a130  
SQL> SELECT name FROM v$datafile ORDER BY 1;  
  
NAME  
-----  
-----  
/u01/app/oracle/oradata/CDBDEV/B29964A7B1066D26E0536310ED0A031  
2/datafile/o1_mf_s  
ysaux_hsgbrmkx_.dbf  
  
/u01/app/oracle/oradata/CDBDEV/B29964A7B1066D26E0536310ED0A031  
2/datafile/o1_mf_s  
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 GUID, 888B23FF74E42ED6E0530100007F6226 in the path name.

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

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

```

SQL> SELECT tablespace_name, contents FROM dba_tablespaces;

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

SQL>

```

- e. Exit SQL*Plus.

```

SQL> EXIT
...
[oracle@edvmr1p0 ~]$

```

5. Run the `dropCDBDEV.sh` script to clean up your environment.

```

[oracle@edvmr1p0 ~]$
/home/oracle/labs/DBMod_CreateDB/dropCDBDEV.sh
[oracle@edvmr1p0 ~]$

```

6. Close the terminal window.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

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

Overview

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

For Instructor Use Only.
This document should not be distributed.

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.

Tasks

1. Open a terminal window as the `oracle` OS user and source the `oraenv` script to set the environment to `orclcdb`.

```
[oracle@edvmr1p0 ~]$ . oraenv
ORACLE_SID = [CDBDEV] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 ~]$
```

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

```
/home/oracle/labs/DBMod_CreateDB/setup_orclpdb3.sh
```

Note: Ignore any errors for unable to drop objects.

```
[oracle@edvmr1p0 ~]$
/home/oracle/labs/DBMod_CreateDB/setup_orclpdb3.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
'/u01/app/oracle/oradata/ORCLCDB/orclpdb3': File exists

SQL> CREATE PLUGGABLE DATABASE ORCLPDB3
  2  ADMIN USER admin IDENTIFIED BY WELCOME123##  ROLES=(CONNECT)
```

```
3
FILE_NAME_CONVERT=(' /u01/app/oracle/oradata/ORCLCDB/pdbseed',' /u0
1/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 cloud_4U;

User created.

SQL> grant dba to test;

Grant succeeded.

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

Table created.

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

```
7 /
PL/SQL procedure successfully completed.

SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
...
SQL>
```

- Shut down the database instance in the `IMMEDIATE` mode. `Normal` is the default shutdown mode if no mode is specified. During the `IMMEDIATE` mode of shutdown, no new connections are allowed, user 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 the `normal` mode, the database instance waits for all users to disconnect before completing the shutdown; however, **no** new connections are allowed. When a shutdown is issued in a session, control will not be returned to the session until shutdown is complete.

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

- Show the current user.

Note: SQL*Plus is still running, and the current user is `SYS`.

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

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

```
SQL> SHOW CON_NAME
ERROR:
ORA-01034: ORACLE not available
Process ID: 0
Session ID: 0 Serial number: 0
Help: https://docs.oracle.com/error-help/db/ora-01034/

SP2-1545: This feature requires Database availability.
Help: https://docs.oracle.com/error-help/db/sp2-1545/

SQL>
```

7. Start up the database instance in the `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 the `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
SQL>
```

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

```
SQL> ALTER DATABASE MOUNT;

Database altered.

SQL>
```

9. Open the database in the default `READ/WRITE` mode by using the `ALTER DATABASE OPEN` 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 and are opened. When the database is open, all users can access the database instance.

```
SQL> ALTER DATABASE OPEN;

Database altered.

SQL>
```

10. Show the current container name.

```
SQL> SHOW CON_NAME

CON_NAME
-----
CDB$ROOT

SQL>
```

11. Show the current user.

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

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

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

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

SQL>
```

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

```
SQL> COLUMN con_name FORMAT A16
SQL> SELECT con_id, con_name, state FROM DBA_PDB_SAVED_STATES;

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

SQL>
```

14. Remove the saved states.

- a. Set the pluggable databases closed.

```
SQL> ALTER PLUGGABLE DATABASE all CLOSE;

Pluggable database altered.

SQL>
```

- b. Set the saved states.

```
SQL> ALTER PLUGGABLE DATABASE all SAVE STATE;  
  
Pluggable database altered.  
  
SQL>
```

- c. View the PDB states.

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

15. Verify the behavior of the saved states:

- a. Shut down the CDB.

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

- b. Start up the CDB.

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

- c. View the actual states of the PDBs.

```
SQL> SELECT con_id, name, open_mode FROM v$pdbs;  
  
CON_ID NAME          OPEN_MODE  
----- -----  
      2 PDB$SEED      READ ONLY  
      3 ORCLPDB1      MOUNTED  
      4 ORCLPDB2      MOUNTED  
      5 ORCLPDB3      MOUNTED  
  
SQL>
```

16. Reset the PDB saved states to open:

- Set the pluggable databases to open.

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

```
Pluggable database altered.
```

```
SQL>
```

- Set the saved states for all PDBs.

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

```
Pluggable database altered.
```

```
SQL>
```

- View the saved states.

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

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

```
SQL>
```

17. Shut down a single PDB, ORCLPDB3.

```
SQL> ALTER PLUGGABLE DATABASE orclpdb3 close immediate;
```

```
Pluggable database altered.
```

```
SQL>
```

18. View the status of the PDBs.

```
SQL> SHOW PDBS

  CON_ID CON_NAME          STATE
  -----
    2  PDB$SEED      READ ONLY
    3  ORCLPDB1      OPEN
    4  ORCLPDB2      OPEN
    5  ORCLPDB3      MOUNTED

SQL>
```

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

```
SQL> DROP PLUGGABLE DATABASE orclpdb3 INCLUDING DATAFILES;

Pluggable database dropped.

SQL>
```

20. Exit SQL*Plus.

```
SAL> EXIT
...
[oracle@edvmr1p0 ~] $
```

21. Close all terminals.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 6: Managing Database Instances

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 6-1: Investigating Initialization Parameter Files

Overview

In this practice, you will 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. Open a terminal window as the `oracle` OS user and use `oraenv` to set the environment variables for the `orclcdb` database.

```
[oracle@edvmr1p0 ~]$ . oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
...
SQL>
```

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

```
SQL> SHOW PARAMETER SPFILE

NAME                                     TYPE        VALUE
-----
-----
spfile                               string
/u01/app/oracle/product/23.4.0/dbhome_1/dbs/spfileorclcdb.ora
SQL>
```

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

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

```
SQL> HOST
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/dbs
[oracle@edvmr1p0 dbs]$ ls
hc_CDBDEV.dat    hc_orclcdb.dat   lkORCLCDB      spfileorclcdb.ora
hc_CDBTEST.dat   init.ora       orapworclcdb
[oracle@edvmr1p0 dbs]$
```

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

- c. Use the `cat` or `more` command to view the contents of the sample text initialization parameter file (PFILE), `init.ora`. Then exit from the HOST shell back to SQL*Plus.

```
[oracle@edvmr1p0 dbs]$ cat init.ora
#
# $Header: dbms/admin/init.ora /main/25 2015/05/14 15:02:30
kasingha Exp $
#
# Copyright (c) 1991, 2015, Oracle and/or its affiliates. All
rights reserved.
# NAME
#   init.ora
# FUNCTION
# NOTES
# MODIFIED
#   kasingha  05/12/15 - 21041456 - fix copyright 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    07/07/93 - add more comments regarded archive
start
```

```

#      maporter   10/29/92 - Add vms_sga_use_gblpagfile=TRUE
#      jloaiza    03/07/92 - change ALPHA to BETA
#      danderso   02/26/92 - change db_block_cache_protect to
_db_block_cache_p
#      ghallmar   02/03/92 - db_directory -> db_domain
#      maporter   01/12/92 - merge changes from branch 1.8.308.1
#      maporter   12/21/91 - bug 76493: Add control_files
parameter
#      wbridge    12/03/91 - use of %c in archive format is
discouraged
#      ghallmar   12/02/91 - add global_names=true,
db_directory=us.acme.com
#      thayes     11/27/91 - Change default for cache_clone
#      jloaiza    08/13/91 -           merge changes from branch
1.7.100.1
#      jloaiza    07/31/91 -           add debug stuff
#      rlim       04/29/91 -           removal of char_is_varchar2
#      Bridge     03/12/91 - log_allocation no longer exists
#      Wijaya     02/05/91 - remove obsolete parameters
#
#####
#####
# Example INIT.ORA file
#
# This file is provided by Oracle Corporation as a starting
point for
# customizing the Oracle Database installation for your site.
#
# NOTE: The values that are used in this file are example values
only.
# You may want to adjust those values for your specific
requirements.
# You might also consider using the Database Configuration
Assistant
# tool (DBCA) to create a server-side initialization parameter
file
# and to size your initial set of tablespaces. See the
# Oracle Database 2 Day DBA guide for more information.
#####
#####

# Change '<ORACLE_BASE>' to point to the oracle base (the one
you specify at
# install time)

```

```

db_name='ORCL'
memory_target=1G
processes = 150
audit_file_dest='<ORACLE_BASE>/admin/orcl/adump'
audit_trail ='db'
db_block_size=8192
db_domain=''
db_recovery_file_dest='<ORACLE_BASE>/fast_recovery_area'
db_recovery_file_dest_size=2G
diagnostic_dest='<ORACLE_BASE>'
dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
# You may want to ensure that control files are created on
separate physical
# devices
control_files = (ora_control1, ora_control2)
compatible ='11.2.0'

[oracle@edvmr1p0 dbs]$ exit

```

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

```

SQL> CREATE pfile='$ORACLE_HOME/dbs/initorclcdb.ora' FROM
spfile;

File created
SQL> !
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/dbs
[oracle@edvmr1p0 dbs]$

```

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

```

[oracle@edvmr1p0 dbs]$ cat initorclcdb.ora
orclcdb._data_transfer_cache_size=0
orclcdb._db_cache_size=788529152
orclcdb._inmemory_ext_roarea=0
orclcdb._inmemory_ext_rwarea=0
orclcdb._inmemory_size=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=1056964608
orclcdb.__streams_pool_size=16777216
orclcdb.__unified_pga_pool_size=0
orclcdb._instance_recovery_bloom_filter_size=1048576
*.compatible='23.0.0'
*.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=16446m
*.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'
[oracle@edvmr1p0 dbs]$
```

f. Return to SQL*Plus.

```
[oracle@edvmr1p0 dbs]$ exit
SQL>
```

5. If the database server doesn't find an `SPFILE`, then the text initialization parameter file will be used. Now you'll set up a test to see how the search works when you start the database instance.

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

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

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

```
SQL> !
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/dbs
[oracle@edvmr1p0 dbs]$
```

- d. Rename the `spfileorclcdb.ora` file to `spfileorclcdb.ora_original`.

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

```
[oracle@edvmr1p0 dbs]$ mv spfileorclcdb.ora
spfileorclcdb.ora_original
[oracle@edvmr1p0 dbs]$
```

- e. Return to SQL*Plus.

```
[oracle@edvmr1p0 dbs]$ exit
SQL>
```

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

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

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

```
SQL> SHOW PARAMETER spfile
```

NAME	TYPE	VALUE
spfile	string	

```
SQL>
```

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

```
SQL> SHUTDOWN IMMEDIATE
```

```
Database closed.  
Database dismounted.  
ORACLE instance shut down.
```

```
SQL>
```

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

```
SQL> host
```

```
[oracle@edvmr1p0 ~]$
```

- Change to the \$ORACLE_HOME/dbs directory.

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/dbs
```

```
[oracle@edvmr1p0 dbs]$
```

- Rename the spfileorclcdb_original.ora file to spfileorclcdb.ora.

```
[oracle@edvmr1p0 dbs]$ mv spfileorclcdb.ora_original  
spfileorclcdb.ora
```

```
[oracle@edvmr1p0 dbs]$
```

- Return to SQL*Plus.

```
[oracle@edvmr1p0 dbs]$ exit
```

```
SQL>
```

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

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

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

```
SQL> SHOW PARAMETER spfile
NAME                                     TYPE        VALUE
-----
spfile                               string
/u01/app/oracle/product/23.4.0/dbhome_1/dbs/spfileorclcdb.ora
SQL>
```

7. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~] $
```

Practice 6-2: Viewing Initialization Parameters by Using SQL*Plus

Overview

In this practice, you will view initialization parameters (parameters) by using SQL*Plus. You will 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 the basic parameters by using the SHOW PARAMETER command. The basic parameters are those parameters that you are likely to modify.

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

```
[oracle@edvmr1p0 ~]$ . oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
...
SQL>
```

3. View the values of the `DB_NAME` and `DB_DOMAIN` parameters. Together, these values create the global database name.

- a. View the value of the `DB_NAME` parameter. This parameter specifies the current database identifier. If you have multiple databases, the value of this parameter should match the Oracle instance identifier of each one to avoid confusion with other databases running on the system.

```
SQL> SHOW PARAMETER db_name
```

NAME	TYPE	VALUE
---	---	---

db_name	string	orclcdb
SQL>		

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

SQL> SHOW PARAMETER db_domain		
NAME	TYPE	VALUE
db_domain	string	
SQL>		

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

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

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

SQL> SHOW PARAMETER db_recovery_file_dest		
NAME	TYPE	VALUE
db_recovery_file_dest	string	/u01/app/oracle/fast_recovery_area
db_recovery_file_dest_size	big integer	16446M
SQL>		

5. View the SGA_TARGET and SGA_MAX_SIZE parameters.

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

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

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

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

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

```
SQL> SHOW PARAMETER sga

NAME                      TYPE        VALUE
-----
allow_group_access_to_sga  boolean     FALSE
lock_sga                  boolean     FALSE
pre_page_sga               boolean     TRUE
sga_max_size               big integer 1920M
sga_min_size               big integer 0
sga_target                 big integer 1920M

SQL>
```

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

<code>SQL> SHOW PARAMETER undo_tablespace</code>		
NAME	TYPE	VALUE
<code>undo_tablespace</code>	string	<code>UNDOTBS1</code>
<code>SQL></code>		

- 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 is installed.

<code>SQL> SHOW PARAMETER compatible</code>		
NAME	TYPE	VALUE
<code>compatible</code>	<code>string</code>	<code>23.0.0</code>
<code>noncdb_compatible</code>	boolean	<code>FALSE</code>
<code>SQL></code>		

- 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.

<code>SQL> SHOW PARAMETER control_files</code>		
NAME	TYPE	VALUE
<code>control_files</code>	string	<code>/u01/app/oracle/oradat a/ORCLCDB/control01.ct l, /u01/app/oracle/fas t_recovery_area/ORCLCD B/control02.ctl</code>
<code>SQL></code>		

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

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

```
SQL> SHOW PARAMETER sessions

NAME                      TYPE        VALUE
-----
java_max_sessionspace_size integer     0
java_soft_sessionspace_limit integer    0
license_max_sessions       integer     0
license_sessions_warning   integer     0
sessions                  integer   472
shared_server_sessions     integer

SQL>
```

View Advanced Parameters:

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

10. View the TRANSACTIONS parameter. This is an advanced parameter and seldom needs any adjustment. This parameter specifies how many rollback segments should be brought online when the UNDO_MANAGEMENT initialization parameter is equal to MANUAL. A transaction is assigned to a rollback segment when the transaction starts, and it can't change for the life of the transaction. A transaction table exists in the rollback segment header with limited space, limiting how many transactions a single segment can support. Therefore, X number of concurrent transactions require at least Y number of rollback segments. With Oracle Automatic Undo Management, the database creates rollback segments, brings them online, takes them offline, and drops them as needed.

```
SQL> SHOW PARAMETER transactions

NAME                      TYPE        VALUE
-----
-
transactions               integer    519
transactions_per_rollback_segment integer   5

SQL>
```

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

```
SQL> SHOW PARAMETER db_files

NAME                      TYPE        VALUE
-----
db_files                  integer    200

SQL>
```

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

```
SQL> SHOW PARAMETER commit_logging
```

NAME	TYPE	VALUE
commit_logging	string	

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

```
SQL> SHOW PARAMETER commit_wait
```

NAME	TYPE	VALUE
commit_wait	string	

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

```
SQL> SHOW PARAMETER shared_pool_size
```

NAME	TYPE	VALUE
shared_pool_size	big integer	0

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

```
SQL> SHOW PARAMETER db_block_size
```

NAME	TYPE	VALUE
db_block_size	integer	8192

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

```
SQL> SHOW PARAMETER db_cache_size
```

NAME	TYPE	VALUE
db_cache_size	big integer	0

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

```
SQL> SHOW PARAMETER undo_management
```

NAME	TYPE	VALUE
undo_management	string	AUTO

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

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

- a. View the `MEMORY_TARGET` parameter.

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

- b. View the `MEMORY_MAX_TARGET` parameter.

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

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

SQL> SHOW PARAMETER pga_aggregate_target		
NAME	TYPE	VALUE
-----	-----	-----
pga_aggregate_target	big integer	640M
SQL>		

Query Views for Parameter Values:

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

20. Query the data dictionary to find views that contain the word "parameter." The query below returns 68 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$PARAMETER
V$PARAMETER2
...
V$SPPARAMETER
V$SYSTEM_PARAMETER
V$SYSTEM_PARAMETER2
V$SYSTEM_RESET_PARAMETER

68 rows selected.

SQL>
```

21. Explore the V\$PARAMETER view. This view displays the current parameter values in the current session.
- a. View the columns in the V\$PARAMETER view by using the DESCRIBE command. This command returns column names, whether null values are allowed (NOT NULL is displayed if the value cannot be null), and column data types. The results below contain a column named ISSYS_MODIFIABLE. This column is important because it tells you whether a parameter is static or dynamic. If its value is FALSE, then the parameter is static; otherwise it's dynamic. To change a static parameter, you must shut down and restart the database; however, you can modify a dynamic parameter in real time while the database is online.

SQL> DESC v\$parameter		
Name	Null?	Type
NUM		NUMBER
NAME		VARCHAR2 (80)
TYPE		NUMBER
VALUE		VARCHAR2 (4000)
DISPLAY_VALUE		VARCHAR2 (4000)
DEFAULT_VALUE		VARCHAR2 (255)
ISDEFAULT		VARCHAR2 (9)
ISSES_MODIFIABLE		VARCHAR2 (5)
ISSYS_MODIFIABLE		VARCHAR2 (9)
ISPDB_MODIFIABLE		VARCHAR2 (5)
ISINSTANCE_MODIFIABLE		VARCHAR2 (5)
ISMODIFIED		VARCHAR2 (10)
ISADJUSTED		VARCHAR2 (5)
ISDEPRECATED		VARCHAR2 (5)
ISBASIC		VARCHAR2 (5)
DESCRIPTION		VARCHAR2 (255)
UPDATE_COMMENT		VARCHAR2 (255)
HASH		NUMBER
CON_ID		NUMBER

SQL>

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

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

Optional: Before entering the following command, you can enter SET PAUSE ON to cause a pause after each page output. Press Enter to display each next page. After all the 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                  FALSE    FALSE
active_instance_count        FALSE
...
transactions                 FALSE    519
transactions_per_rollback_segment FALSE    5
...
wallet_root                  FALSE
workarea_size_policy         IMMEDIATE AUTO
xml_db_events                IMMEDIATE enable

530 rows selected.

SQL>
```

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

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

```
SQL> SELECT name, value FROM v$parameter
      WHERE name LIKE '%pool%';

NAME                                VALUE
-----
shared_pool_size                      0
large_pool_size                       0
java_pool_size                        0
streams_pool_size                     0
shared_pool_reserved_size             73987522
memoptimize_pool_size                 0
buffer_pool_keep                     
buffer_pool_recycle                  
olap_page_pool_size                  0

9 rows selected.

SQL>
```

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

```
SQL> DESC v$spparameter

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

SQL>
```

- 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

535 rows selected.

SQL>
```

23. Explore the `V$PARAMETER2` view. This view contains information about the initialization parameters that are currently in effect for the session. For parameters with more than one value assigned such as the `control_files` parameter, each parameter value will be listed as a row in the view's output. A new session inherits parameter values from the instance-wide values displayed in the `V$SYSTEM_PARAMETER2` view.

- a. View the columns in the `V$PARAMETER2` view by using the `DESCRIBE` command.

```
SQL> DESC v$parameter2

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

SQL>
```

- b. Query NAME and VALUE in the V\$PARAMETER2 view. Browse the rows returned by the query.

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

NAME                                VALUE
-----
...
shrd_dupl_table_refresh_rate        60
multishard_query_data_consistency   strong
multishard_query_partial_results    not allowed

535 rows selected.

SQL>
```

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

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

```
SQL> DESC v$system_parameter

Name          Null?    Type
-----
NUM           NUMBER
NAME          VARCHAR2 (80)
TYPE          NUMBER
VALUE         VARCHAR2 (4000)
DISPLAY_VALUE VARCHAR2 (4000)
DEFAULT_VALUE VARCHAR2 (255)
ISDEFAULT    VARCHAR2 (9)
ISSES_MODIFIABLE VARCHAR2 (5)
ISSYS_MODIFIABLE VARCHAR2 (9)
ISPDB_MODIFIABLE VARCHAR2 (5)
ISINSTANCE_MODIFIABLE VARCHAR2 (5)
ISMODIFIED   VARCHAR2 (8)
ISADJUSTED   VARCHAR2 (5)
ISDEPRECATED VARCHAR2 (5)
ISBASIC      VARCHAR2 (5)
DESCRIPTION   VARCHAR2 (255)
UPDATE_COMMENT VARCHAR2 (255)
HASH          NUMBER
CON_ID        NUMBER

SQL>
```

- b. Query NAME and VALUE in the V\$SYSTEM_PARAMETER view. Browse the rows returned by the query.

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

NAME	VALUE
common_user_prefix	
multishard_query_data_consistency	strong
multishard_query_partial_results	not allowed

```
629 rows selected.
```

```
SQL>
```

25. Exit SQL*Plus.

```
SQL> exit
```

```
...
```

```
[oracle@edvmr1p0 ~]$
```

Practice 6-3: Modifying Initialization Parameters by Using SQL*Plus

Overview

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

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

Assumptions

- You are connected to the compute node as the `oracle` user.
- The Oracle environment is set to access the `orclcldb` instance.

Tasks

Modify a Session-Level Parameter:

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

1. Open a terminal window as the `oracle` OS user and start SQL*Plus and log in to the database as the `sys` user with the `SYSDBA` privilege.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

- 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

SQL>
```

- 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

SQL>
```

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

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

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

Session altered.

SQL>
```

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

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

LAST_NAME          HIRE_DATE
-----
King                17-JUN-03
Kochhar              21-SEP-05
...
Higgins              07-JUN-02
Gietz                07-JUN-02

107 rows selected.

SQL>
```

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

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

Session altered.

SQL>
```

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

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

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

107 rows selected.

SQL>
```

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

```
SQL> SHOW PARAMETER nls_date_format

NAME                      TYPE        VALUE
-----
nls_date_format           string      mon dd yyyy

SQL>
```

8. Exit from `ORCLPDB1` to end your session.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

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

- a. Start `SQLPLUS` by using Easy Connect syntax:

```
sqlplus system/WElcome123##@//localhost:1521/orclpdb1
```

Note: In this exercise, we use `localhost` as `<full hostname>` since we are using a local host.

```
[oracle@edvmr1p0 ~]$ sqlplus
system/WElcome123##@//localhost:1521/orclpdb1
...
SQL>
```

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

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

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0) (PORT=1521)
))

SQL>
```

- c. Connect to the root container as `sysdba`.

```
SQL> CONNECT / as sysdba
...
SQL>
```

- d. Query the V\$SERVICES view to discover the network name.

Note: The network name is the name of the fully qualified database instance and is formed by default from 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
SQL>
```

- e. Using values from steps 9(a) through 9(d), 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.

Note: The '//' characters preceding the <full hostname> are optional. In this exercise, we use localhost as <full hostname> since we are using a local host.

```
SQL> CONNECT system/WElcome123##@//localhost:1521/orclpdb1
Connected.
SQL>
```

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

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

LAST_NAME          HIRE_DATE
-----
...
Higgins           07-JUN-02
Gietz             07-JUN-02

107 rows selected.

SQL>
```

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

```
SQL> SHOW PARAMETER nls_date_format
```

NAME	TYPE	VALUE
nls_date_format	string	

```
SQL>
```

Modify a Dynamic System-Level Parameter:

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

1. Connect to the root container with the `SYSDBA` privilege. If you try to update the `JOB_QUEUE_PROCESSES` parameter from `ORCLPDB1`, you'll get an error. Also, you'll need the `SYSDBA` privilege to restart the database instance later.

```
SQL> CONNECT / as sysdba
...
SQL>
```

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

```
SQL> 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

SQL>
```

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

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

System altered.

SQL>

- 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
<code>job_queue_processes</code>	integer	15
<code>max_datapump_jobs_per_pdb</code>	integer	AUTO
<code>max_datapump_parallel_per_job</code>	string	AUTO

SQL>

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

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

```
SQL> SHUTDOWN IMMEDIATE
```

Database closed.

Database dismounted.

ORACLE instance shut down.

SQL>

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

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

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

```
SQL> SHOW PARAMETER job
NAME                                     TYPE        VALUE
-----
job_queue_processes                      integer     15
max_datapump_jobs_per_pdb               integer     AUTO
max_datapump_parallel_per_job           string      AUTO
SQL>
```

Modify a Static System-Level Parameter:

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

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

```
SQL> COL name FORMAT A30
SQL> SELECT name, isses_modifiable, issys_modifiable, value
FROM v$parameter WHERE name = 'sec_max_failed_login_attempts';

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

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

```
SQL> ALTER SYSTEM SET sec_max_failed_login_attempts = 2
      SCOPE=BOTH;
ALTER SYSTEM SET sec_max_failed_login_attempts = 2 SCOPE=BOTH
*
ERROR at line 1:
ORA-02095: specified initialization parameter cannot be
modified
Help: https://docs.oracle.com/error-help/db/ora-02095/

SQL>
```

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

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

System altered.

SQL>
```

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

```
SQL> SHOW PARAMETER sec_max

NAME                                     TYPE        VALUE
-----
sec_max_failed_login_attempts           integer     3

SQL>
```

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

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

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

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

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

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

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

```
SQL> 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.
SQL>
```

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

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

```
System altered.
```

```
SQL>
```

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

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

Practice 6-4: Viewing Diagnostic Information

Overview

In this practice, you will 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 `orclcldb` 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. Open a terminal window as the `oracle` OS user; start SQL*Plus and log in to the database as the `sys` user with the `SYSDBA` privilege.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

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

```

SQL> COL name FORMAT A23
SQL> COL value FORMAT A55
SQL> SELECT name, value FROM v$diag_info;

NAME          VALUE
-----
Diag Enabled   TRUE
ADR Base       /u01/app/oracle
ADR Home       /u01/app/oracle/diag/rdbms/orclcdb/orclcdb
Diag Trace     /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace
Diag Alert     /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert
Diag Incident   /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/incident
Diag Cdump     /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/cdump
Health Monitor /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/hm
Default Trace File /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace/
                    orclcdb_ora_8778.trc

Active Problem Count  0
Active Incident Count 0
ORACLE_HOME          /u01/app/oracle/product/19.3.0/dbhome_1
Attention Log         /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace/
                    attention_orclcdb.log

13 rows selected.

SQL>

```

3. Exit SQL*Plus.

```

SQL> EXIT
...
[oracle@edvmr1p0 ~]$

```

View the Alert Log:

1. View the XML version of the alert log. The `log.xml` file is the XML version of the alert log.
 - a. Browse to the `/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert` directory.

```
[oracle@edvmr1p0 ~]$ cd
/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert
[oracle@edvmr1p0 alert]$
```

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

```
[oracle@edvmr1p0 alert]$ ls -l
total 2200
-rw-r----- 1 oracle oinstall 2246772 Oct 16 15:13 log.xml
[oracle@edvmr1p0 alert]$
```

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

```
[oracle@edvmr1p0 alert]$ 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$ROOT'>
<txt>ALTER SYSTEM SET sec_max_failed_login_attempts=3
SCOPE=SPFILE;
</txt>
</msg>
[oracle@edvmr1p0 alert]$
```

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

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

```
[oracle@edvmr1p0 alert]$ cd
/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace
[oracle@edvmr1p0 trace]$
```

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

```
[oracle@edvmr1p0 trace]$ ls -l *log
-rw-r----- 1 oracle oinstall 278430 Feb 14 22:22
alert_orclcdb.log
-rw-r----- 1 oracle oinstall     2971 Feb 12 03:33
attention_orclcdb.log
-rw-r----- 1 oracle oinstall     2520 Feb 14 22:18 drcorclcdb.log
[oracle@edvmr1p0 trace]$
```

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

```
[oracle@edvmr1p0 trace]$ tail -100 alert_orclcdb.log
...
=====
Dumping current patch information
=====
No patches have been applied
=====
2024-02-14T22:19:14.255420+00:00
db_recovery_file_dest_size of 16446 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.
2024-02-14T22:19:14.772489+00:00
Setting Resource Manager plan
SCHEDULER:DEFAULT_MAINTENANCE_PLAN via scheduler window
2024-02-14T22:22:05.606373+00:00
ALTER SYSTEM SET sec_max_failed_login_attempts=3 SCOPE=SPFILE;
2024-02-14T22:22:48.776821+00:00
ALTER SYSTEM SET sec_max_failed_login_attempts=3 SCOPE=SPFILE;

[oracle@edvmr1p0 trace]$
```

- d. Change your directory to the home directory.

```
[oracle@edvmr1p0 trace]$ cd
[oracle@edvmr1p0 ~]$
```

Use ADRCI to View the Alert Log:

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

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

```
[oracle@edvmr1p0 ~]$ adrci
ADRCI: Release 23.0.0.0.0 - Limited Production on Wed Feb 14
22:54:58 2024

Copyright (c) 1982, 2023, Oracle and/or its affiliates. All
rights reserved.

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

2. View the alert log by using the SHOW ALERT command. The show alert command will prompt you for which alert log file to display, unless you are in the database's diagnostic directory. Choose the alert log for the orclcdb database:

Note: The alert log file is 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
```

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

```
...
space available in the underlying filesystem or ASM diskgroup.
Setting Resource Manager plan SCHEDULER:DEFAULT_MAINTENANCE_PLAN
via scheduler window
2024-02-14 22:22:05.606000 +00:00
ALTER SYSTEM SET sec_max_failed_login_attempts=3 SCOPE=SPFILE;
2024-02-14 22:22:48.776000 +00:00
ALTER SYSTEM SET sec_max_failed_login_attempts=3 SCOPE=SPFILE;
```

4. Enter **/Starting ORACLE/** and press return. Press **N** (uppercase) to search from the bottom of the file to find the last time the instance was started. The following will be similar to your alert log.

Note: Here lowercase and uppercase are important because `vi` distinguishes them, unless you ignore them by setting: `set ic`.

```
...
2024-02-12 05:52:50.950000 +00:00
Starting ORACLE instance (normal) (OS id: 1670)
*****
Sys-V shared memory will be used for creating SGA
*****
*****
Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
    Per process system memlock (soft) limit = 128G
...
```

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

```
...
ALTER DATABASE MOUNT
Control File SGA cache allocated 8388608 bytes.
...
```

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

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

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

```
adrci > exit
[oracle@edvmr1p0 ~] $
```

Log DDL Statements in the DDL Log File:

1. Determine if DDL logging is enabled in **ORCLPDB1**. If not, enable it by setting the value for the **ENABLE_DDL_LOGGING** initialization parameter to **TRUE**.
 - a. Start **SQL*Plus** and log in to the database as the **sys** user with the **SYSDBA** privilege.

```
[oracle@edvmr1p0 ~] $ sqlplus / as sysdba
...
SQL>
```

- b. Switch to **ORCLPDB1**.

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

SQL>
```

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

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

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

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

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

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

3. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ cd  
/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/log  
[oracle@edvmr1p0 log]$
```

5. List the contents of the log directory.

```
[oracle@edvmr1p0 log]$ ls  
ddl  ddl_orclcdb.log  debug  debug_orclcdb.log  hcs  
hcs_orclcdb.log  imbd  test  
[oracle@edvmr1p0 log]$
```

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

```
[oracle@edvmr1p0 log]$ cat ddl_orclcdb.log
2024-02-14T23:36:16.017202+00:00
diag_adl:CREATE TABLE test (name varchar2(15))
2024-02-14T23:36:28.077662+00:00
diag_adl:DROP TABLE test
[oracle@edvmr1p0 log]$
```

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

```
[oracle@edvmr1p0 log]$ cd ddl
[oracle@edvmr1p0 ddl]$ ls
log.xml
[oracle@edvmr1p0 ddl]$ cat log.xml
<msg time='2024-02-14T23:36:16.017+00:00' org_id='oracle'
comp_id='rdbsm'
msg_id='kpdbLogDDL:24048:2946163730' type='UNKNOWN'
group='diag_adl'
level='16' host_id='edvmr1p0' host_addr='10.237.16.202'
...
[oracle@edvmr1p0 ddl]$
```

8. Close the terminal window.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 7: Oracle Net Services Overview

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 8: Configuring Naming Methods

Overview

In these practices, you will configure your network environment.

For Instructor Use Only.
This document should not be distributed.

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

Overview

In this practice, you will:

- Configure your network environment.
- Use local naming and create a new network service name called TESTORCL.
- Test your network changes by attempting to connect to the other database by using the TESTORCL service name.

Tasks

1. Open a terminal window as the oracle OS user. Make a copy of your tnsnames.ora file. It is in your database \$ORACLE_HOME/network/admin directory.
 - a. Change the directory to \$ORACLE_HOME/network/admin and then list your current working directory.

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/network/admin  
[oracle@edvmr1p0 admin]$ pwd  
/u01/app/oracle/product/23.4.0/dbhome_1/network/admin  
[oracle@edvmr1p0 admin]$
```

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

```
[oracle@edvmr1p0 admin]$ cp tnsnames.ora tnsnames.old  
[oracle@edvmr1p0 admin]$
```

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

```
[oracle@edvmr1p0 admin]$ 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.old  
[oracle@edvmr1p0 admin]$
```

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

```
[oracle@edvmr1p0 admin]$ hostname -f  
edvmr1p0.us.oracle.com  
[oracle@edvmr1p0 admin]$
```

3. Create the TESTORCL net service for ORCLCDB.

- a. Open tnsnames.ora.

```
[oracle@edvmr1p0 admin]$ vi tnsnames.ora
```

- b. Add an entry for the new service as shown below and then save your changes.

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

4. Test your changes to the network configuration by using SQL*Plus and connecting as a system user.

- a. Invoke SQL*Plus and connect by using the testorcl service name and then show the container name you are connected to using SHOW con_name.

```
[oracle@edvmr1p0 admin]$ sqlplus system/WELCOME123##@testorcl
...
SQL> SHOW con_name

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

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

```
SQL> EXIT
...
[oracle@edvmr1p0 admin]$
```

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

Overview

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

Tasks

1. Open a terminal window as the `oracle` OS user and locate and view your local `tnsnames.ora` file before you add a net service name to it.
 - a. Change the directory to `$ORACLE_HOME/network/admin`.

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/network/admin  
[oracle@edvmr1p0 admin]$
```

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

Note: Your output may be slightly different.

```
[oracle@edvmr1p0 admin]$ ls -l  
total 20  
-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  
[oracle@edvmr1p0 admin]$
```

2. Create a net service name, `MYPDB1`, for `ORCLPDB1`.

- a. Open `tnsnames.ora`.

```
[oracle@edvmr1p0 admin]$ vi tnsnames.ora
```

- b. Add an entry for the new service and save your changes.

```
...
MYPDB1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
edvmr1p0.us.oracle.com)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = orclpdb1)
    )
  )
)
```

3. Verify that the MYPDB1 entry has been added to the tnsnames.ora file.

```
[oracle@edvmr1p0 admin]$ cat tnsnames.ora
...
MYPDB1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST =
edvmr1p0.us.oracle.com)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orclpdb1)
    )
...
[oracle@edvmr1p0 admin]$
```

4. 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 (orclpdb1) is available.

```
[oracle@edvmr1p0 admin]$ tnsping mypdb1

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on
16-OCT-2020 22:52:48
Copyright (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 (0 msec)
[oracle@edvmr1p0 admin]$
```

5. Connect to ORCLPDB1 and verify the container.

- Start SQL*Plus and connect to ORCLPDB1 as the system user by using the MYPDB1 net service name.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@MyPDB1
...
SQL>
```

- Verify that the current container name is ORCLPDB1.

```
SQL> SHOW con_name

CON_NAME
-----
ORCLPDB1

SQL>
```

6. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 admin]$
```

7. Close the terminal.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 9: Configuring and Administering the Listener

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 9-1: Exploring the Default Listener

Overview

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

Tasks

1. Open a terminal window as the `oracle` OS user; start SQL*Plus and log in as the `sys` user with the `SYSDBA` privilege.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

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

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

- b. `SERVICE_NAMES`: This parameter identifies the service names that users can use in their connection strings to connect to the database instance. By default, the service name takes on the same name as the global database name, `orclcdb.example.com`, which is a combination of the `DB_NAME` parameter (`orclcdb`) and the `DB_DOMAIN` parameter (`example.com`). If the `DB_DOMAIN` parameter is blank so will be 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>
```

- c. LOCAL_LISTENER: This parameter specifies the alias names for local listeners that resolve to addresses in the `tnsnames.ora` file (or other address repository as configured for your system). If there are multiple aliases, they must be separated by commas and all values enclosed by one set of double quotation marks. The results show one alias, `LISTENER_ORCLCDB` (`LISTENER_<SID>`). Keep in mind that this isn't the name of the listener. It's an alias for it.

```
SQL> SHOW PARAMETER local_listener

NAME                      TYPE        VALUE
-----
local_listener            string      LISTENER_ORCLCDB

SQL>
```

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

```
SQL> SHOW PARAMETER remote_listener

NAME                      TYPE        VALUE
-----
remote_listener           string      null

SQL>
```

3. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

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

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

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/network/admin
[oracle@edvmr1p0 admin]$
```

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

```
[oracle@edvmr1p0 admin]$ ls
listener.ora  samples  shrept.lst  tnsnames.old  tnsnames.ora
[oracle@edvmr1p0 admin]$
```

- c. View the `tnsnames.ora` file by using the `less` command (case matters). The entry for the `LISTENER_ORCLCDB` alias contains one protocol address, which consists of a host name, a port number (1521, which is the default port number), and protocol (TCP). The protocol address is the listener's "end point". A listener's end point does not contain a listener name or a `CONNECT_DATA` section like the `ORCLPDB1` and `ORCLCDB` entries.

Note: `less` uses `vi` like key commands to move about the file. End the `less` session with '`q`'.

NOTE: The content of your file may differ from that given in the below screenshot.

```
[oracle@edvmr1p0 admin]$ less tnsnames.ora
...
LISTENER_ORCLCDB =
    (ADDRESS = (PROTOCOL = TCP) (HOST = edvmr1p0) (PORT = 1521))

ORCLCDB =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST =
edvmr1p0.us.oracle.com) (PORT = 1521))
        )
        (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = orclcdb)
        )
    )
...
[oracle@edvmr1p0 admin]$
```

5. 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:

- The `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.
- The `listener.ora` file in the `$ORACLE_HOME/network/admin` directory
- Naming method, for example, a `tnsnames.ora` file

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

```
[oracle@edvmr1p0 admin]$ cat listener.ora
# listener.ora Network Configuration File:
/u01/app/oracle/product/23.4.0/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

[oracle@edvmr1p0 admin]$
```

6. 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`.

```
[oracle@edvmr1p0 admin]$ lsnrctl

LSNRCTL for Linux: Version 23.0.0.0.0 - Limited Availability on
15-FEB-2024 01:31:35

Copyright (c) 1991, 2023, Oracle. All rights reserved.

Welcome to LSNRCTL, type "help" for information.

LSNRCTL>
```

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

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

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

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

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

```
LSNRCTL> show current_listener
Current Listener is LISTENER
LSNRCL>
```

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

```
LSNRCTL> status

LSNRCTL for Linux: Version 23.0.0.0.0 - Limited Availability
on 15-FEB-2024 22:07:18

Copyright (c) 1991, 2023, Oracle. All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0) (PORT=1521)
))
```

```
STATUS of the LISTENER
-----
Alias                      LISTENER
Version                   TNSLSNR for Linux: Version
23.0.0.0.0 - Limited Availability
Start Date                12-FEB-2024 05:52:47
Uptime                    3 days 16 hr. 14 min. 30 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/oracle/product/23.4.0/dbhome_1/network/admin/listener
                           .ora
Listener Log File         /u01/app/oracle/diag/tnslnsr/edvmr1p0/listener/alert/log.xml
Listening Endpoints Summary...
                              

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0) (PORT=1521)
))
Services Summary...
Service "0a625dce1978a21ce063eadf5e64ef08" has 1 instance(s).
    Instance "orclcdb", status READY, has 1 handler(s) for this
service...
Service "10fa8a666314ba14e0636910ed0acd5a" has 1 instance(s).
    Instance "orclcdb", status READY, has 1 handler(s) for this
service...
Service "10fa8bb51487bae9e0636910ed0acd86" 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>
```

Note: 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.

- d. 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

LSNRCTL for Linux: Version 23.0.0.0.0 - Limited Availability
on 15-FEB-2024 22:10:36

Copyright (c) 1991, 2023, Oracle. All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=edvmr1p0)(PORT=1521))
)
Services Summary...
Service "0a625dce1978a21ce063eadf5e64ef08" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "DEDICATED" established:2 refused:0 state:ready
      LOCAL SERVER
Service "10fa8a666314ba14e0636910ed0acd5a" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "DEDICATED" established:2 refused:0 state:ready
      LOCAL SERVER
Service "10fa8bb51487bae9e0636910ed0acd86" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "DEDICATED" established:2 refused:0 state:ready
      LOCAL SERVER
Service "orclcdb" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "DEDICATED" established:2 refused:0 state:ready
      LOCAL SERVER
Service "orclcdbXDB" has 1 instance(s).
  Instance "orclcdb", 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: 89209>
```

```
(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=1088
5))
Service "orclpdb1" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "DEDICATED" established:2 refused:0 state:ready
      LOCAL SERVER
Service "orclpdb2" has 1 instance(s).
  Instance "orclcdb", status READY, has 1 handler(s) for this
service...
  Handler(s):
    "DEDICATED" established:2 refused:0 state:ready
      LOCAL SERVER
The command completed successfully
LSNRCTL>
```

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

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

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

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

```
LSNRCTL> exit
...
[oracle@edvmr1p0 admin]$
```

Practice 9-2: Creating a Second Listener

Overview

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

Assumptions

- You are logged in as the `oracle` user.

Tasks

1. Open the `tnsnames.ora` file and create an entry that resolves a LISTENER2 alias to a protocol address.

- a. Open a terminal window as the `oracle` OS user and obtain your host name and domain. The format is `host.domain`.

```
[oracle@edvmr1p0 ~]$ hostname -f  
edvmr1p0.us.oracle.com  
[oracle@edvmr1p0 ~]$
```

- b. Change to the `$ORACLE_HOME/network/admin` directory.

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/network/admin  
[oracle@edvmr1p0 admin]$
```

- c. Copy the `tnsnames.ora` file to `tnsnames.ora.old2`.

```
[oracle@edvmr1p0 admin]$ cp tnsnames.ora tnsnames.ora.old2  
[oracle@edvmr1p0 admin]$
```

- d. 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, and save your changes. You can copy and paste `LISTENER_ORCLCDB` as a starting point. Specify your host and domain for the host name discovered in step 1(b), 1561 for the port number, and TCP as the protocol. The `vi` editor is shown here.

```
[oracle@edvmr1p0 admin]$ vi tnsnames.ora  
...  
# tnsnames.ora Network Configuration File:  
/u01/app/oracle/product/23.4.0/dbhome_1/network/admin/tnsnames.ora  
# Generated by Oracle configuration tools.
```

```

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

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

```

2. Modify the LOCAL_LISTENER initialization parameter to include both LISTENER_ORCLCDB and LISTENER2 aliases.
 - a. Open a new terminal window as the oracle OS user; start SQL*Plus and log in as the sys user with the SYSDBA privilege.

```
[oracle@edvmr1p0 admin]$ sqlplus / as sysdba
...
SQL>
```

- b. View the LOCAL_LISTENER initialization parameter. The LISTENER_ORCLCDB value 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>
```

- c. 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>
```

- d. 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>
```

- e. Confirm that LISTENER_ORCLCDB and LISTENER2 are values for the LOCAL_LISTENER initialization parameter and exit SQL*Plus.

Note: The output was formatted for viewing.

```
SQL> SHOW PARAMETER local_listener

NAME          TYPE        VALUE
-----
local_listener    string    LISTENER_ORCLCDB,
                                LISTENER2

SQL> EXIT
[oracle@edvmr1p0 admin]$
```

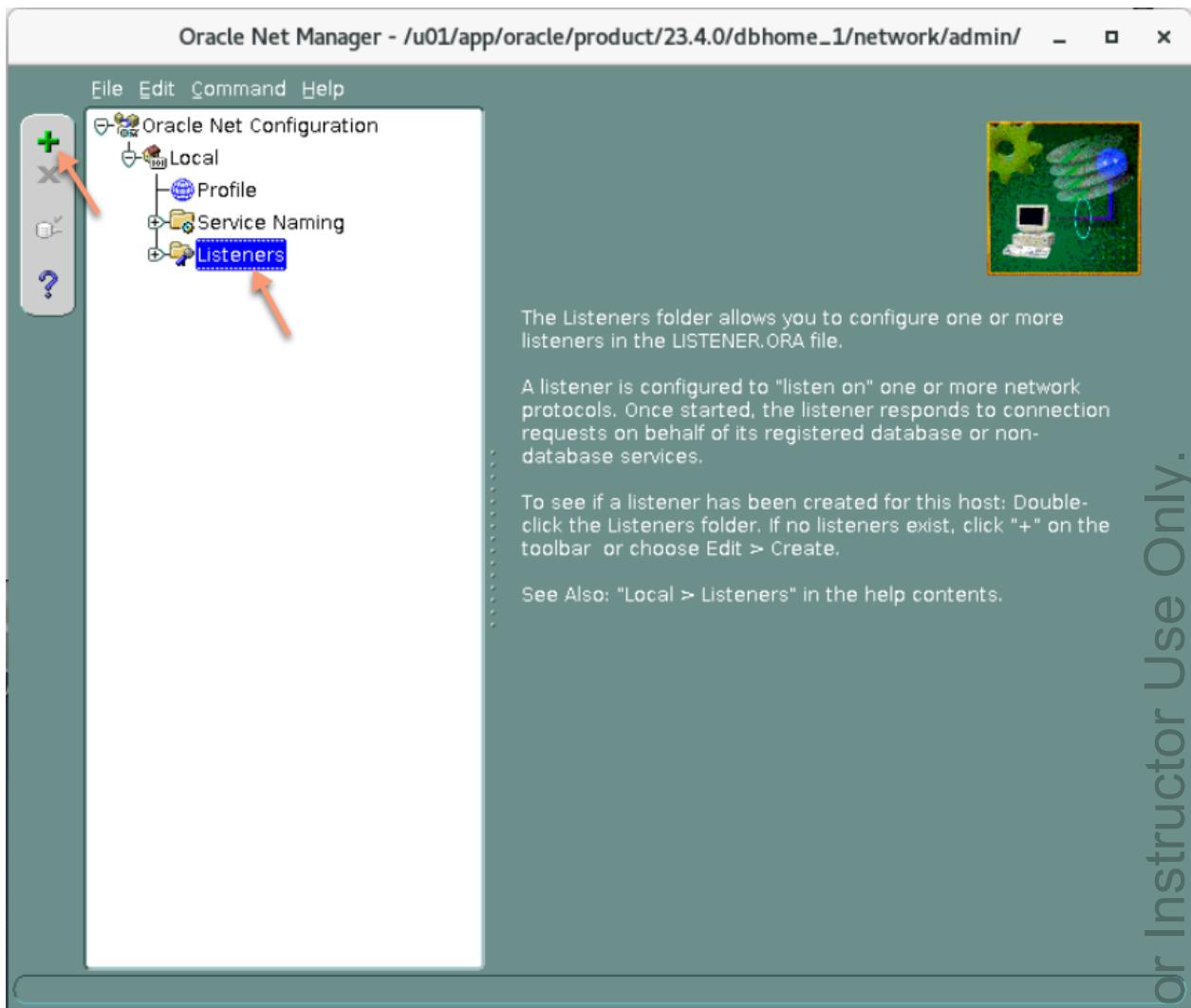
- 3. To manage LISTENER2 with the Listener Control utility, you need to add an entry in the listener.ora file so that the utility knows how to connect to LISTENER2. In this task, you will use Oracle Net Manager to add the entry. It's important to remember that dynamic service registration does not make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

- a. Make a copy of the listener.ora file; call it listener.old

```
[oracle@edvmr1p0 admin]$ cp listener.ora listener.old
[oracle@edvmr1p0 admin]$
```

- b. In the terminal window, start Oracle Net Manager. Use the netmgr command.

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



- d. In the Choose Listener Name dialog box, enter **LISTENER2** and click **OK**. LISTENER2 is added to the list of listeners.
- e. With **LISTENER2** selected on the left side, click **Add Address**.
- f. In the Address1 panel on the right, leave Listening Locations selected in the drop-down list, leave TCP/IP selected as the protocol, and set the host name to **your host and domain**. In the Port box, enter **1561**.
- g. For interest, select **General Parameters** from the drop-down list. Review the configuration options on the General, Logging & Tracing, and Authentication tabs.
- h. Select **Database Services** from the drop-down list. There are no databases currently configured for the listener.
- i. Select **File** and then **Save Network Configuration**.
- j. Select **File** and then **Exit** to exit Oracle Net Manager. You just added an entry into the **listeners.ora** file.

- k. 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.

```
[oracle@edvmr1p0 admin]$ cat listener.ora
# listener.ora Network Configuration File:
/u01/app/oracle/product/23.4.0/dbhome_1/network/admin/listener
.ora
# Generated by Oracle configuration tools.

LISTENER =
  (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

[oracle@edvmr1p0 admin]$
```

4. Using the Listener Control utility, check the status of LISTENER2.

- a. In the terminal window, start the Listener Control utility.

```
[oracle@edvmr1p0 admin]$ lsnrctl
...
LSNRCTL>
```

- b. Check the status of LISTENER2 by issuing the status command. Your results indicate "no listener" and "Connection refused" because you have just created the listener and need to start it.

```
LSNRCTL> status LISTENER2
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0) (PORT=1561)
))
TNS-12541: Cannot connect. No listener at host edvmr1p0 port
1561.
TNS-12560: Database communication protocol error.
TNS-00511: No listener
Linux Error: 111: Connection refused
LSNRCTL>
```

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

```
LSNRCTL> start LISTENER2
Starting /u01/app/oracle/product/23.4.0/dbhome_1/bin/tnslsnr:
please wait...

TNSLSNR for Linux: Version 23.0.0.0.0 - Limited Availability
System parameter file is
/u01/app/oracle/product/23.4.0/dbhome_1/network/admin/listener
.ora
Log messages written to
/u01/app/oracle/diag/tnslsnr/edvmr1p0/listener2/alert/log.xml
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.c
om) (PORT=1561)))

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0) (PORT=1561)
))
STATUS of the LISTENER
-----
Alias                      LISTENER2
Version                    TNSLSNR for Linux: Version
23.0.0.0.0 - Limited Availability
Start Date                 15-FEB-2024 21:56:20
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/23.4.0/dbhome_1/network/admin/listener  
.ora  
Listener Log File  
/u01/app/oracle/diag/tnslnsr/edvmr1p0/listener2/alert/log.xml  
Listening Endpoints Summary...  
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=1561)))  
The listener supports no services  
The command completed successfully  
LSNRCTL>
```

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.

- d. 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. Exit out of LSNRCTL.

```
LSNRCTL> status LISTENER2  
Connecting to  
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0) (PORT=1561))  
)  
STATUS of the LISTENER  
-----  
Alias LISTENER2  
Version TNSLSNR for Linux: Version  
23.0.0.0.0 - Limited Availability  
Start Date 15-FEB-2024 21:56:20  
Uptime 0 days 0 hr. 49 min. 14 sec  
Trace Level off  
Security ON: Local OS Authentication  
SNMP OFF  
Listener Parameter File  
/u01/app/oracle/product/23.4.0/dbhome_1/network/admin/listener  
.ora  
Listener Log File  
/u01/app/oracle/diag/tnslnsr/edvmr1p0/listener2/alert/log.xml  
Listening Endpoints Summary...  
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=edvmr1p0.us.oracle.com) (PORT=1561)))  
Services Summary...  
Service "0a625dce1978a21ce063eadf5e64ef08" has 1 instance(s).  
Instance "orclcdb", status READY, has 1 handler(s) for this service...  
Service "10fa8a666314ba14e0636910ed0acd5a" has 1 instance(s).  
Instance "orclcdb", status READY, has 1 handler(s) for this service...
```

```

Service "10fa8bb51487bae9e0636910ed0acd86" 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> exit
[oracle@edvmr1p0 admin]$

```

- Using Easy Connect syntax, start SQL*Plus and connect to the CDB using LISTENER2. Make sure to specify the non-default port number 1561.

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.

```

[oracle@edvmr1p0 admin]$ sqlplus
system/WElcome123##@localhost:1561/orclcdb
...
SQL>

```

- Reset the LOCAL_LISTENER parameter.

```

SQL> ALTER SYSTEM SET local_listener='LISTENER_ORCLCDB' ;
System altered.

SQL>

```

- Exit SQL*Plus.

```

SQL> exit
...
[oracle@edvmr1p0 ~]$

```

8. In the terminal window, start Oracle Net Manager to remove LISTENER2.
 - a. Use **netmgr** command.
 - b. In the Oracle Net Manager navigation pane, expand **Local** and then **Listeners**. Select **LISTENER2** and click the red x sign to remove the selected listener. Click **Yes** to confirm.
 - c. Select **File** and then **Save Network Configuration**.
 - d. Select **File** and then **Exit** to exit Oracle Net Manager.
9. Close all open terminal windows.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 10: Configuring a Shared Server Architecture

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 10-1: Configuring Shared Server Mode

Overview

In this practice, you will configure a shared server mode.

Tasks

1. Open a terminal window as the oracle OS user and log in to the CDB with SYSDBA privileges.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

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

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

```
SQL> SHOW PARAMETER shared_server  
  
NAME                      TYPE        VALUE  
-----  
max_shared_servers          integer  
shared_server_sessions      integer  
shared_servers              integer      1  
  
SQL>
```

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

```
SQL> SHOW PARAMETER dispatchers  
  
NAME                      TYPE        VALUE  
-----  
--  
dispatchers                string     (PROTOCOL=TCP)  
(SERVICE=orclcdbXDB)  
max_dispatchers             integer  
  
SQL>
```

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 such as HTTP and FTP require a shared server. This results in a SHARED_SERVER value of 1. Although a shared server is enabled, this configuration permits only sessions that connect to the XDB service to use the shared server. To enable a 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.

3. Enable three shared servers in your database.

```
SQL> ALTER SYSTEM SET shared_servers = 3;  
  
System altered.  
  
SQL>
```

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

```
SQL> SHOW PARAMETER dispatchers  
  
NAME          TYPE    VALUE  
----  
--  
dispatchers    string  (PROTOCOL=TCP)  
(SERVICE=orclcdbXDB)  
max_dispatchers integer  
  
SQL>
```

Question: Do you need to configure any additional dispatchers?

Answer: Yes. When a 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.

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

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

6. Confirm the change:

```
SQL> SHOW PARAMETER dispatchers

NAME          TYPE    VALUE
-----
dispatchers   string  (PROTOCOL=TCP)
max_dispatchers integer

SQL>
```

7. Exit SQL*Plus.

```
SQL> exit
...
[oracle@edvmr1p0 admin]$
```

8. Close all the terminals.

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

Overview

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

Tasks

1. **(Catchup Step)** If you *skipped Practice 10-1*, run the `setup_lab10-2.sh` script to catch up. Otherwise, proceed to **step 2**.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_NetSvcs/setup_lab10-2.sh  
...  
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ cd $ORACLE_HOME/network/admin  
[oracle@edvmr1p0 admin]$ pwd  
/u01/app/oracle/product/23.4.0/dbhome_1/network/admin  
[oracle@edvmr1p0 admin]$
```

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

```
[oracle@edvmr1p0 admin]$ cp tnsnames.ora tnsnames.ora.10-2  
[oracle@edvmr1p0 admin]$
```

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

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

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

3. Invoke SQL*Plus and connect to the database using a dispatcher service.

Note: Call this **Terminal 1**.

```
[oracle@edvmr1p0 admin]$ sqlplus system/WELCOME123##@test_ss
...
SQL>
```

4. Open another terminal window called **Terminal 2**. Log in to SQL*Plus as a user with SYSDBA privileges.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
...
SQL>
```

5. 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
00000009F54D420	00	00000009F967A18	NONE

```
SQL>
```

6. In **Terminal 1**, log out of the SQL*Plus session that uses the `test_ss` service.
7. 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>
```

8. Log out of SQL*Plus and close both terminal windows.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 11: Creating PDBs from Seed

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 11-1: 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 data files to the `/u01/app/oracle/oradata/` directory.

Tasks

1. Open a terminal as the `oracle` OS user and run the

```
/home/oracle/labs/DBMod_PDBs/setup_tns.sh
```

```
[oracle@edvmr1p0 ~]$ /home/oracle/labs/DBMod_PDBs/setup_tns.sh  
The Oracle base remains unchanged with value /u01/app/oracle  
[oracle@edvmr1p0 ~]$
```

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

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

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

- b. Execute the `CREATE PLUGGABLE DATABASE` command.

```
SQL> CREATE PLUGGABLE DATABASE orclpdb3  
ADMIN USER pdb3_admin IDENTIFIED BY WELCOME123##  
ROLES=(CONNECT)  
CREATE_FILE_DEST='/u01/app/oracle/oradata';  
2 3 4  
Pluggable database created.  
SQL>
```

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

```
SQL> COLUMN pdb_name FORMAT A16
SQL> SELECT pdb_id, pdb_name, status FROM cdb_pdbs;

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

SQL>
```

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

```
SQL> ALTER PLUGGABLE DATABASE orclpdb3 OPEN;
Pluggable database altered.

SQL> SELECT pdb_id, pdb_name, status FROM cdb_pdbs;

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

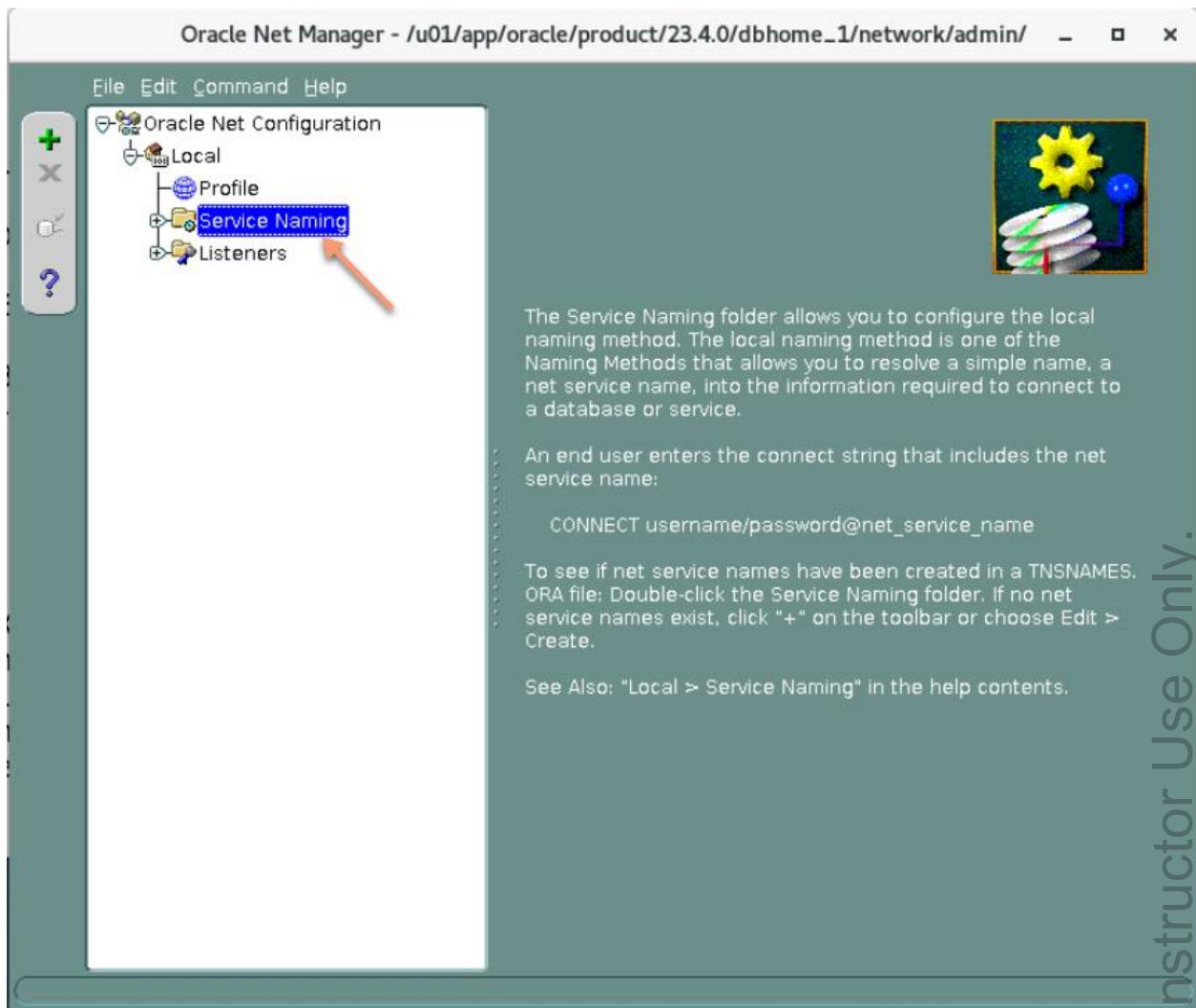
SQL> EXIT
```

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

- a. Launch `netmgr`.

```
[oracle@edvmr1p0 ~]$ netmgr
```

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



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

6. Connect to the PDB and verify that the data files are in the correct location. 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 `/database name/PDB_ID`.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WElcome123##@PDB3
...
SQL> SELECT name FROM v$logfile;

NAME
-----
-----
-----
/u01/app/oracle/oradata/ORCLCDB/11754F6EA06FDEEFE0631511ED0AF93C
/datafile/o1_mf_system_lwx8wm58_.dbf
/u01/app/oracle/oradata/ORCLCDB/11754F6EA06FDEEFE0631511ED0AF93C
/datafile/o1_mf_sysaux_lwx8wm5o_.dbf
/u01/app/oracle/oradata/ORCLCDB/11754F6EA06FDEEFE0631511ED0AF93C
/datafile/o1_mf_undotbs1_lwx8wm5p_.dbf

SQL>
```

7. Verify that the service is ORCLPDB3 and then exit from SQL*Plus.

```
SQL> COL name FORMAT A15
SQL> SELECT name FROM v$services;

NAME
-----
orclpdb3

SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

8. Exit the terminal.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 12: Using Other Techniques to Create PDBs

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 13: Managing PDBs

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 13-1: Renaming a PDB

Overview

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

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. Open a terminal window as oracle OS user and execute the

```
$HOME/labs/DBMod_PDBs/setup_pdb3.sh shell script to create ORCLPDB3 in  
ORCLCDB.
```

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_PDBs/setup_orclpdb3.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. Set the environment to ORCLCDB, connect to the root container as SYSDBA, and list the PDBs.

Note: The con_id values may vary from the output shown below.

```
[oracle@edvmr1p0 ~]$ . oraenv  
ORACLE_SID = [orclcdb] ? orclcdb  
The Oracle base has been set to /u01/app/oracle  
[oracle@edvmr1p0 ~]$ 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>
```

3. Change the global database name for ORCLPDB3 to PDB3_ORCL.

```
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-65045: pluggable database not in a restricted mode
Help: https://docs.oracle.com/error-help/db/ora-65045/
SQL>
```

4. Close ORCLPDB3.

```
SQL> ALTER PLUGGABLE DATABASE orclpdb3 CLOSE IMMEDIATE;

Pluggable database altered.

SQL>
```

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

```
SQL> ALTER PLUGGABLE DATABASE orclpdb3 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>
```

6. Change the global database name for ORCLPDB3 to PDB3_ORCL.

```
SQL> ALTER PLUGGABLE DATABASE orclpdb3 RENAME global_name TO
      pdb3_orcl;
ALTER PLUGGABLE DATABASE orclpdb3 RENAME global_name TO pdb3_orcl
*
Pluggable database altered.

SQL>
```

7. Close PDB3_ORCL and reopen it in normal read write mode

```
SQL> ALTER PLUGGABLE DATABASE pdb3_orcl CLOSE IMMEDIATE;
Pluggable database altered.

SQL> ALTER PLUGGABLE DATABASE pdb3_orcl OPEN;
Pluggable database altered.

SQL>
```

8. Verify that PDB3_ORCL is in READ WRITE mode, and then exit sqlplus.

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

CON_ID NAME          OPEN_MODE   RES
----- -----
...
3  PDB3_ORCL        READ WRITE NO

SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

9. Add the PDB3_ORCL service name to the tnsnames.ora file with the script
\$HOME/labs/DBMod_PDBs/add_pdb3_tns.sh.

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_PDBs/add_pdb3_orcl_tns.sh
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 ~]$
```

10. Test the connection to PDB3_ORCL and exit when complete.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@pdb3_orcl
...
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

Practice 13-2: Setting Parameter Values for PDBs

Overview

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

Tasks

- Not all initialization parameters are modifiable at the PDB level. A modifiable one, OPTIMIZER_USE_SQL_PLAN_BASELINES, has been chosen for the example to show how initialization parameters behave at the PDB and CDB level.

Open a terminal window connect to `orclcldb` as `SYSDBA` and validate that the parameter is PDB modifiable.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
Connected.
SQL> SELECT ispdb_modifiable FROM v$parameter
      WHERE name = 'optimizer_use_sql_plan_baselines';

ISPDB
-----
TRUE

SQL>
```

- Check the current value of `OPTIMIZER_USE_SQL_PLAN_BASELINES`.

```
SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines

NAME                           TYPE        VALUE
-----
optimizer_use_sql_plan_baselines    boolean     TRUE

SQL>
```

- Connect to `ORCLPDB1` PDB in `ORCLCDB` and check the current value of `OPTIMIZER_USE_SQL_PLAN_BASELINES`.

```
SQL> CONNECT sys/WELCOME123##@orclpdb1 as sysdba
Connected.
SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines

NAME                           TYPE        VALUE
-----
optimizer_use_sql_plan_baselines    boolean     TRUE

SQL>
```

4. Change the parameter value to FALSE in ORCLPDB1 and validate the change.

```
SQL> ALTER SYSTEM SET optimizer_use_sql_plan_baselines=false
scope=both;

System altered.

SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines

NAME                      TYPE        VALUE
-----
optimizer_use_sql_plan_baselines    boolean      FALSE

SQL>
```

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

- a. Connect to the CDB create PDB named test, and open it.

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

SQL> CREATE PLUGGABLE DATABASE test ADMIN USER admin
      IDENTIFIED BY WELCOME123## roles=(connect)
      create_file_dest='/u01/app/oracle/oradata/ORCLCDB/test';
2   3

Pluggable database created.

SQL> ALTER PLUGGABLE DATABASE test OPEN;

Pluggable database altered.

SQL>
```

- b. 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>
```

- c. Connect to the test PDB as SYSBDA and view the value for the parameter optimizer_use_sql_plan_baselines.

```
SQL> CONNECT sys/WELCOME123##@test as sysdba
Connected.
SQL> SHOW PARAMETER optimizer_use_sql_plan_baselines
SHOW PARAMETER optimizer_use_sql_plan_baselines
NAME                                     TYPE      VALUE
-----
optimizer_use_sql_plan_baselines        boolean   TRUE
SQL>
```

6. Connect to ORCLPDB1 as SYSDBA, close and open the PDB and view the value for the parameter optimizer_use_sql_plan_baselines.

```
SQL> CONNECT sys/WELCOME123##@orclpdb1 as sysdba
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>
```

Note: The value remained unchanged after reopening the PDB.

7. Check the parameter value after CDB shutdown/startup in both the CDB root and PDBs.
 - a. Connect to the CDB as SYSDBA shut down and restart the database instance, open all the PDBs, and validate the value of the `optimizer_use_sql_plan_baselines` for all the containers.

Note: Your output of `SHOW PDBS` command may vary from the values shown below.

```

SQL> CONNECT / as sysdba
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
...
Database mounted.
Database opened.
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;

Pluggable database altered.

SQL> COL value FORMAT A30
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 PDB3_ORCL         READ WRITE NO
  6 TEST               READ WRITE NO

SQL> SELECT con_id, value FROM v$system_parameter WHERE name
='optimizer_use_sql_plan_baselines';

CON_ID VALUE
-----
  0 TRUE
  3 FALSE

SQL>

```

Note: The above query will only return the value of the parameter for those containers that have explicitly set the parameter at that container level

- Drop PDBs, TEST and PDB3_ORCL (or ORCLPDB3) if exists.

Note: you may not have PDB3_ORCL (or ORCLPDB3) if you *skipped* the previous labs.

```
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 PDB3_ORCL    READ WRITE NO
 6 TEST         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;

Pluggable database dropped.

SQL> DROP PLUGGABLE DATABASE pdb3_orcl INCLUDING DATAFILES;

Pluggable database dropped.

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

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

SQL>
```

For Instructor Use Only.
This document should not be distributed.

9. Confirm PDBs were dropped with a `SHOW PDBS` command.

```
SQL> SHOW PDBS

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

SQL>
```

10. Alter the session to `orclpdb1` reset the `optimizer_use_sql_plan_baselines` parameter to TRUE.

```
SQL> ALTER SESSION SET container=orclpdb1;
Session altered.

SQL> ALTER SYSTEM SET optimizer_use_sql_plan_baselines=TRUE
scope=both;
System altered.

SQL>
```

11. Exit SQL*Plus and close all terminal windows.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 14: Database Storage Overview

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 15: Creating and Managing Tablespaces

Overview

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

For Instructor Use Only.
This document should not be distributed.

Practice 15-1: Viewing Tablespace Information

Overview

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

Assumptions

- You are logged in as the `oracle` user.

Tasks

1. Open a new terminal window and connect to ORCLPDB1 as SYSDBA using SQL*Plus.

```
[oracle@edvmr1p0 ~]# sqlplus sys/WElcome123##@orclpdb1 as sysdba
...
Connected to:
...
SQL>
```

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

```
SQL> DESCRIBE dba_tablespaces

Name          Null?    Type
-----
TABLESPACE_NAME      NOT NULL VARCHAR2(30)
BLOCK_SIZE          NOT NULL NUMBER
INITIAL_EXTENT      NUMBER
NEXT_EXTENT          NUMBER
MIN_EXTENTS          NOT NULL NUMBER
...
DEF_CELLMEMORY      VARCHAR2(14)
DEF_INMEMORY_SERVICE VARCHAR2(12)
DEF_INMEMORY_SERVICE_NAME
VARCHAR2(1000)
LOST_WRITE_PROTECT  VARCHAR2(7)
CHUNK_TABLESPACE    VARCHAR2(1)

SQL>
```

3. 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>
```

4. 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>
```

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

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

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

7. 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 YES value, indicating that the tablespace is included in full database backups by using the BACKUP DATABASE RMAN command.
- BIGFILE contains the NO value, indicating that the tablespace is a smallfile tablespace.
- FLASHBACK_ON contains the YES value, indicating that the tablespace participates in FLASHBACK DATABASE operations.

- ENCRYPT_IN_BACKUP contains the null value, 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.
8. 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>
```

9. 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>

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

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

SQL>

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

```
SQL> COL 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 YES value, indicating that the autoextend 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.

12. Find out how many segments are there 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)
-----
          2792
SQL>
```

13. 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>
```

14. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~] $
```

Viewing Tablespace Information by SQL*Developer:

15. Launch SQL*Developer.
16. Expand **orclpdb1** in the Connections pane.
17. In the DBA view, expand **orclpdb1**.

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

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

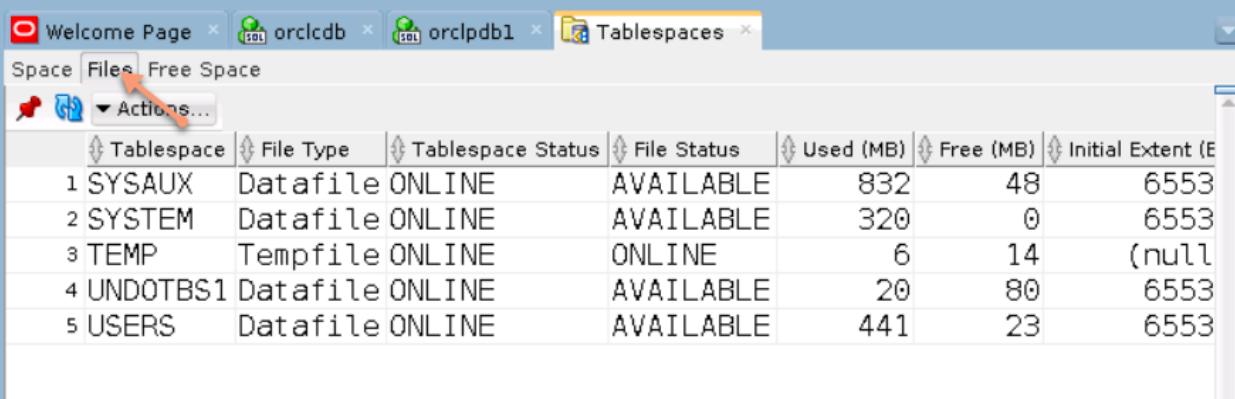
The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : TABLESPACE SYS.null@orclpdb1". The left sidebar has two main sections: "Connections" and "DBA". Under "Connections", there are entries for "Oracle Connections" (with "orclcdb" and "orclpdb1" expanded), "Database Schema Service Connection", and "Database Configuration" (with "orclcdb" and "orclpdb1" expanded). Under "DBA", there are sections for "Database Configuration", "Database Status", "Data Pump", "Performance", "RMAN Backup/Recovery", "Resource Manager", "SQL Translator Framework", "Scheduler", "Security", "Storage" (which is expanded to show "Archive Logs", "Control Files", "Datafiles", "Redo Log Groups", "Rollback Segments", and "Tablespaces"), and "Tables" (which is expanded to show "SYSAUX", "SYSTEM", and "TFMP"). An orange arrow points from the "Tablespaces" section under "Storage" to the "Tablespaces" tab in the top navigation bar. The "Tablespaces" tab is currently selected, displaying a table of tablespace information. The table has columns: Tablespace Name, Allocated (MB), Free (MB), Used (MB), % Free, % Used, and Max. Bytes (MB). The data is as follows:

Tablespace Name	Allocated (MB)	Free (MB)	Used (MB)	% Free	% Used	Max. Bytes (MB)
1 TEMP	20	14	6	70	30	32768
2 UNDOTBS1	100	80	20	80	20	32768
3 SYSTEM	320	0	320	0	100	32768
4 USERS	464	23	441	5	95	32768
5 SYSAUX	880	48	832	5	95	32768

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

The screenshot shows the Oracle SQL Developer interface with the "Space" tab selected in the top navigation bar. The title bar reads "Oracle SQL Developer : TABLESPACE SYS.null@orclpdb1". The left sidebar is identical to the previous screenshot, showing the "Storage" section expanded to include "Tablespaces". An orange arrow points from the "Space" tab in the top navigation bar to the "Space" tab in the table header. The table displays the same data as the previous screenshot, showing tablespace names, sizes, and usage statistics.

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



The screenshot shows the SQL*Developer interface with the 'Tables' tab selected. In the top navigation bar, the 'Tables' tab is highlighted with a red arrow pointing to it. Below the tabs, there are three buttons: a red key icon, a blue folder icon, and a dropdown menu labeled 'Actions...'. The main area is a grid table titled 'Tablespaces' with the following columns: Tablespace, File Type, Tablespace Status, File Status, Used (MB), Free (MB), and Initial Extent (E). The data rows are:

Tablespace	File Type	Tablespace Status	File Status	Used (MB)	Free (MB)	Initial Extent (E)
1 SYSAUX	Datafile	ONLINE	AVAILABLE	832	48	6553
2 SYSTEM	Datafile	ONLINE	AVAILABLE	320	0	6553
3 TEMP	Tempfile	ONLINE	ONLINE	6	14	(null)
4 UNDOTBS1	Datafile	ONLINE	AVAILABLE	20	80	6553
5 USERS	Datafile	ONLINE	AVAILABLE	441	23	6553

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

Question: In this example, how much of the SYSAUX tablespace is used?

Answer: 95 percent of the SYSAUX tablespace has been used. It has 47 MB of free space left.

Note: The values in your database may differ from what are shown in these examples.

22. Close SQL*Developer and all terminal windows.

Practice 15-2: Creating a Tablespace

Overview

In this practice, you will 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 sys 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 can correct the problem.

1. Open a terminal window as the oracle OS user and connect to ORCLPDB1 as the sys user.

```
[oracle@edvmr1p0 ~]$ sqlplus sys/WELCOME123##@orclpdb1 as sysdba  
...  
SQL>
```

2. Execute the

/home/oracle/labs/DBMod_Storage/CreateINVENTORYTablespace.sql script.

```
SQL> SET ECHO ON  
SQL>  
@/home/oracle/labs/DBMod_Storage/CreateINVENTORYTablespace.sql  
  
SQL> CREATE SMALLFILE TABLESPACE INVENTORY  
  2   DATAFILE  
  3     '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF'  
SIZE 5242880  
  4   DEFAULT NOCOMPRESS  
  5   ONLINE  
  6   SEGMENT SPACE MANAGEMENT AUTO  
  7   EXTENT MANAGEMENT LOCAL AUTOALLOCATE;  
  
Tablespace created.  
  
SQL>
```

3. Execute the /home/oracle/labs/DBMod_Storage/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 SYS.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
SQL> --
SQL> -- Create and populate table x
SQL> --
SQL> DECLARE
  2   dummy CHAR(1);
  3   BEGIN
  4     SELECT 'X' INTO DUMMY FROM USER_TABLES WHERE table_name =
'X';
  5     EXECUTE IMMEDIATE 'DROP TABLE X PURGE';
  6   EXCEPTION
  7     WHEN NO_DATA_FOUND THEN
  8       NULL;
  9   END;
10 /
PL/SQL procedure successfully completed.

SQL> SET echo ON
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.

...          <output truncated>

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 increase tablespace INVENTORY by 1MB during
insert or update on table SYS.X
Help: https://docs.oracle.com/error-help/db/ora-01653/

SQL> COMMIT;

Commit complete.

SQL> quit
Disconnected from Oracle Database 23c Enterprise Edition Release
23.0.0.0.0 - Limited Availability
Version 23.4.0.23.11
[oracle@edvmr1p0 ~]$
```

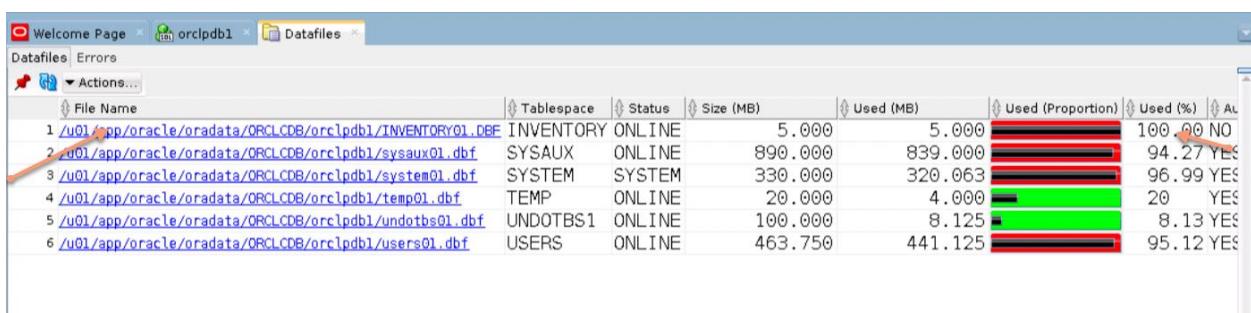
Use SQL*Developer to Increase the Size of the INVENTORY01.DBF Data File:

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.

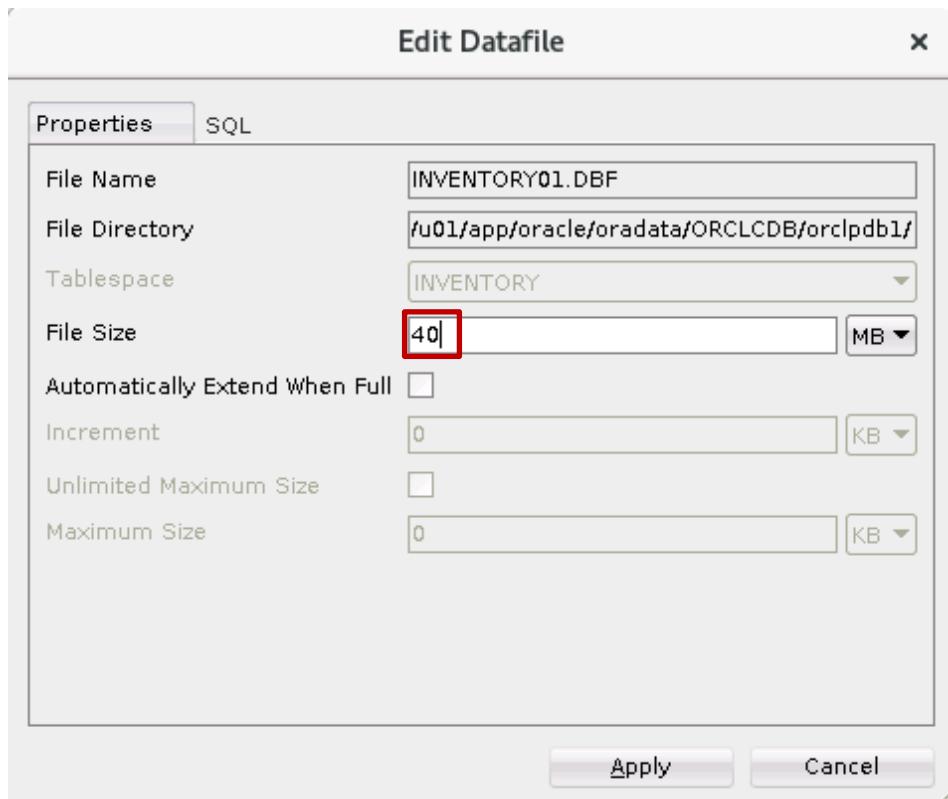
4. Launch SQL*Developer.
5. In the DBA panel, expand **orclpdb1**.
6. Expand **Storage** and then select **Tablespaces**.
7. Double-click the **INVENTORY** tablespace and select the **Datafiles** tab.
8. Now you have found the name of the **INVENTORY01.DBF** data file.



9. In the DBA pane, under **Storage** select **Datafiles**. You can see that the INVENTORY01.DBF data file is used 100 percent.

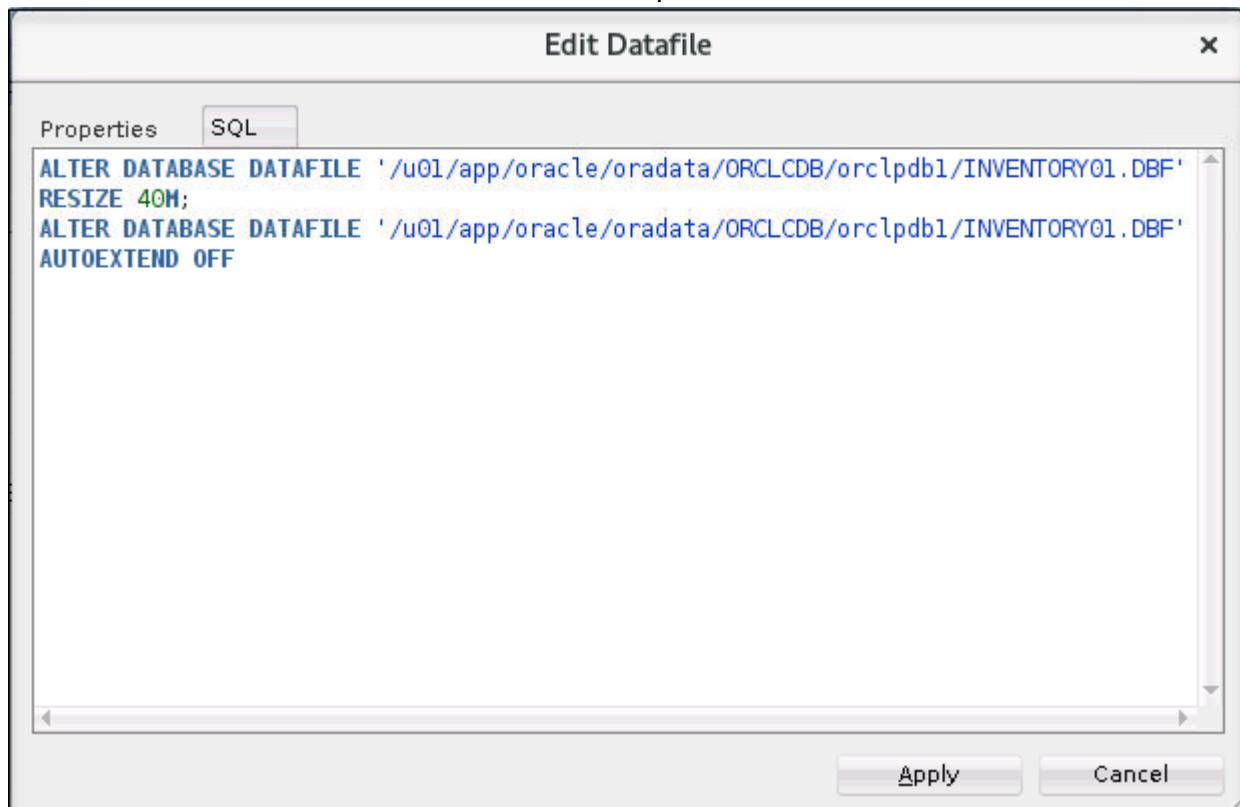


10. Double-click the INVENTORY01.DBF filename. Click **Actions > Edit**.
11. In the Edit Datafile box, enter a file size of **40MB**. Don't click Apply just yet.



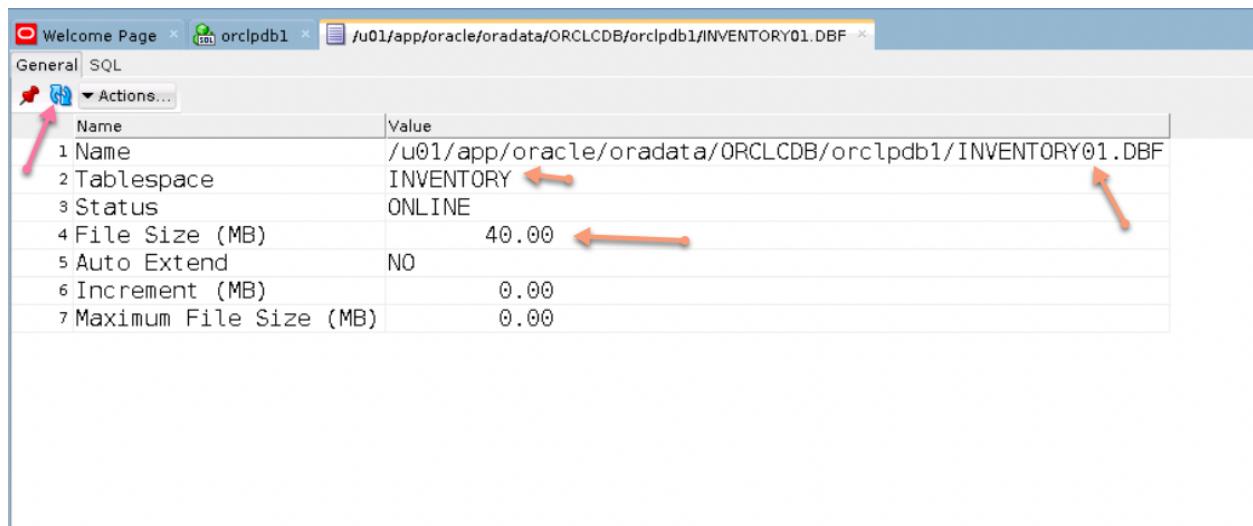
For Instructor Use Only.
This document should not be distributed.

- Click the **SQL** tab to view the SQL command that performs the resize action.



- In the dialog box, click **Apply**.
- In the Successful dialog box, click **OK**. The data file has been successfully resized.
- Verify that the change is reflected in the SQL*Developer interface. The size for the **INVENTORY** tablespace should now be set to 40MB.

Note: To refresh the information in the pane, click the data files in the DBA pane and select the **Inventory** data file again or click the refresh icon.



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

1. In the DBA pane, expand **Storage** and click **Tablespaces**.

The screenshot shows the SQL*Developer interface with the DBA pane open. The title bar says "Welcome Page" and "orclpdb1". The "Tablespaces" tab is selected. A table lists six tablespaces: TEMP, INVENTORY, UNDOTBS1, SYSTEM, USERS, and SYSAUX. The INVENTORY row is highlighted with a blue background. An orange arrow points to the INVENTORY row.

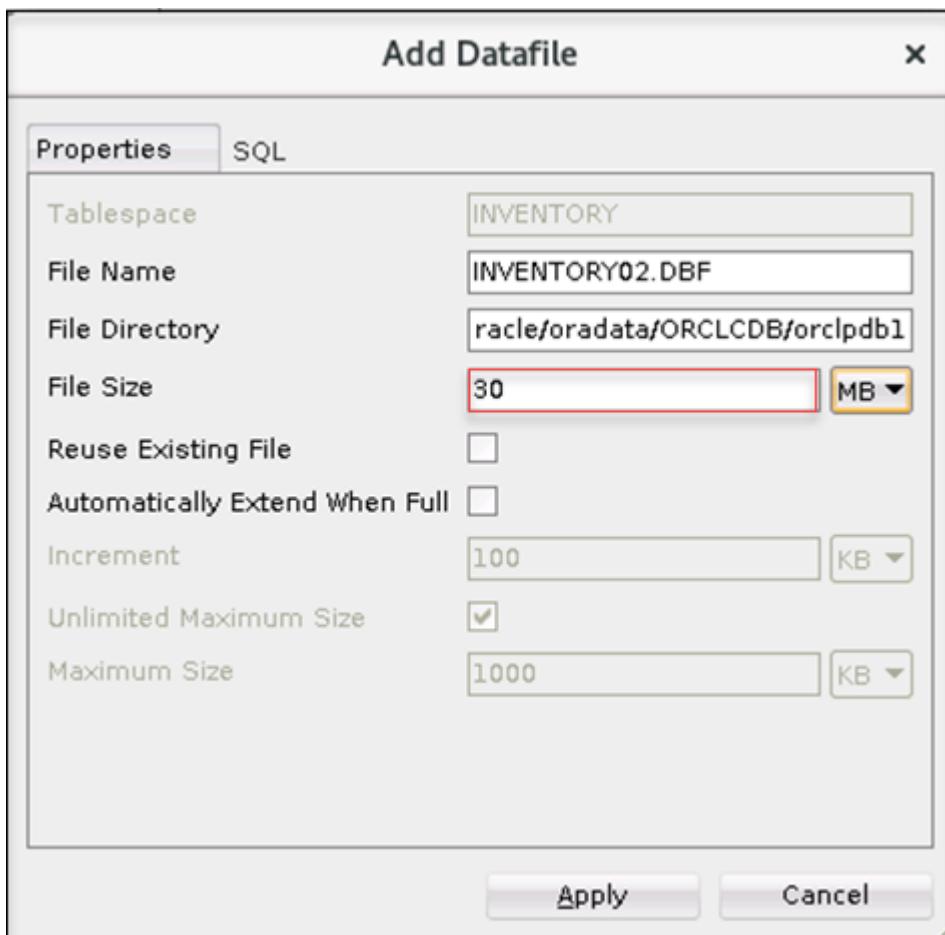
	Tablespace Name	Allocated (MB)	Free (MB)	Used (MB)	% Free	% Used	Max. Bytes (MB)
1	TEMP	20	16	4	80	20	32768
2	INVENTORY	40	35	5	88	12	40
3	UNDOTBS1	100	79	21	79	21	32768
4	SYSTEM	330	10	320	3	97	32768
5	USERS	464	23	441	5	95	32768
6	SYSAUX	890	51	839	6	94	32768

2. Double-click the **INVENTORY** tablespace.
3. Expand **Actions** and then select **Add Datafile**.

The screenshot shows the SQL*Developer interface with the DBA pane open. The title bar says "Welcome Page" and "orclpdb1". The "INVENTORY" tablespace is selected. The "Actions" menu is open, showing options: Edit..., Add Datafile..., Change Storage Management..., Change Read State..., Change Offline/Online..., Drop Tablespace... . An orange arrow points to the "Add Datafile..." option.

	Value
1	INVENTORY
2	8192
3	65536
4	(null)
5 MIN_EXTENTS	1
6 MAX_EXTENTS	2147483645

4. The "Add Datafiles" dialog box is displayed.
 - a. Enter the file name: **INVENTORY02.DBF**.
 - b. Enter file directory: **/u01/app/oracle/oradata/ORCLCDB/orclpdb1/**.
 - c. Enter the file size: **30MB**.



5. Click the **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 the Datafiles subtab and verify that it now has two data files: **INVENTORY01.DBF** and **INVENTORY02.DBF**.

File Name	File ID	Total (MB)	Used (MB)	Free (MB)	Blocks	Autoextensible
1 /u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF	40	40	5	35	5120	NO
2 /u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY02.DBF	42	30	1	29	3840	NO

9. Close the SQL*Developer window.

Use SQL*Plus to Create Table X and Populate It:

As the `sys` 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 have increased the size of the tablespace.

10. Return to your terminal window.
11. Start SQL*Plus and connect to `ORCLPDB1` as the `sys` user.

```
[oracle@edvmr1p0 ~]$ sqlplus sys/WELCOME123##@orclpdb1 as sysdba
...
SQL>
```

12. 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.

... <Output Truncated>

SQL> INSERT INTO x
  2  SELECT * FROM x ;

2048 rows created.

SQL> COMMIT;

Commit complete.

SQL> quit
...
[oracle@edvmr1p0 ~]$
```

13. Start SQL*Plus again and connect to ORCLPDB1 as the sys user.

```
[oracle@edvmr1p0 ~]$ sqlplus sys/WELCOME123##@orclpdb1 as sysdba
...
SQL>
```

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

```
SQL> SELECT table_name FROM all_tables WHERE
  2  tablespace_name='INVENTORY';

TABLE_NAME
-----
X
SQL>
```

Use SQL*Plus to Drop the INVENTORY Tablespace:

15. Drop the INVENTORY tablespace.

```
SQL> DROP TABLESPACE inventory INCLUDING CONTENTS AND DATAFILES;  
  
Tablespace dropped.  
  
SQL>
```

16. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

17. Close the terminal session.

Practice 15-3: Managing Temporary and Permanent Tablespaces

Overview

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

Assumptions

- The PDB, ORCLPDB1, exists and is open.

Tasks

- Open a terminal window as the oracle OS user and then execute \$HOME/labs/DBMod_Storage/glogin_6.sh. This script sets formatting for all the columns selected in the queries.

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_Storage/glogin_6.sh
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@edvmr1p0 ~]$
```

- View the permanent and temporary tablespaces properties in ORCLCDB.

Note:

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
...
SQL> SELECT property_name, property_value
      FROM database_properties
     WHERE property_name like 'DEFAULT_%TABLE%';
2  3
PROPERTY_NAME          PROPERTY_VALUE
-----
DEFAULT_PERMANENT_TABLESPACE USERS
DEFAULT_TEMP_TABLESPACE    TEMP

SQL> SELECT tablespace_name, con_id FROM cdb_tablespaces;

TABLESPACE_NAME CON_ID
-----
SYSTEM          1
SYSAUX          1
UNDOTBS1        1
TEMP            1
USERS           1
SYSTEM          3
SYSAUX          3
```

```

UNDOTBS1          3
TEMP              3
USERS             3
SYSTEM            4
SYSAUX            4
UNDOTBS1          4
TEMP              4
USERS             4

15 rows selected.

SQL> SELECT tablespace_name, con_id FROM cdb_tablespaces
      WHERE tablespace_name like 'TEMP%' ORDER BY 2;
2
TABLESPACE_NAME CON_ID
-----
TEMP           1
TEMP           3
TEMP           4

SQL>
```

3. Create a permanent tablespace CDATA in the CDB root and validate the new tablespace.

```

SQL> CREATE TABLESPACE cdata
      DATAFILE '/u01/app/oracle/oradata/ORCLCDB/cdata_01.dbf'
      SIZE 10m;
2 3
Tablespace created.

SQL> SELECT tablespace_name, con_id FROM cdb_tablespaces
      WHERE tablespace_name = 'CDATA';
2
TABLESPA CON_ID
-----
CDATA        1

SQL>
```

4. Make the CDATA tablespace the default tablespace in the root container and validate the change.

```
SQL> ALTER DATABASE DEFAULT TABLESPACE cdata;

Database altered.

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

SQL>
```

5. Create a permanent tablespace, LDATA, in ORCLPDB1.

```
SQL> CONNECT system/WElcome123##@orclpdb1
Connected.
SQL> CREATE TABLESPACE ldata DATAFILE
  '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/ldata_01.dbf' SIZE 10m;
Tablespace created.

SQL>
```

6. Make the LDATA tablespace the default tablespace in the ORCLPDB1 container and validate the change.

```
SQL> ALTER PLUGGABLE DATABASE DEFAULT TABLESPACE ldata;

Pluggable database altered.

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

7. Create a temporary tablespace in the CDB root.

```
SQL> CONNECT system/WELCOME123##  
Connected.  
SQL> CREATE TEMPORARY TABLESPACE temp_root TEMPFILE  
'/u01/app/oracle/oradata/ORCLCDB/temproot_01.dbf' SIZE 500m;  
  
Tablespace created.  
SQL>
```

8. Make TEMP_ROOT the default temporary tablespace in the CDB root and validate the change.

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

9. Create a temporary tablespace TEMP_PDB1 in ORCLPDB1.

```
SQL> CONNECT system/WELCOME123##@orclpdb1  
Connected.  
SQL> CREATE TEMPORARY TABLESPACE temp_pdb1 TEMPFILE  
'/u01/app/oracle/oradata/ORCLCDB/orclpdb1/temppdb1_01.dbf' SIZE  
100m;  
  
Tablespace created.  
SQL>
```

10. Make TEMP_PDB1 the default temporary tablespace in ORCLPDB1 and validate the change.

```
SQL> ALTER PLUGGABLE DATABASE DEFAULT TEMPORARY TABLESPACE
temp_pdb1;

Pluggable database altered.

SQL> SELECT property_name, property_value
  FROM database_properties
 WHERE property_name like 'DEFAULT_%TABLE%';
 2 3
PROPERTY_NAME          PROPERTY_VALUE
-----
DEFAULT_PERMANENT_TABLESPACE LDATA
DEFAULT_TEMP_TABLESPACE    TEMP_PDB1

SQL>
```

11. Create another temporary tablespace my_temp in ORCLPDB1.

```
SQL> CREATE TEMPORARY TABLESPACE my_temp TEMPFILE
 '/u01/app/oracle/oradata/ORCLCDB/orclpdb1/mytemp_01.dbf' SIZE 10m;

Tablespace created.

SQL>
```

12. Display default tablespaces of a new 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.

Note: You can ignore any error about closing and dropping 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 welcome123## 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/WElcome123##@//localhost:1521/newpdb
Connected.
SQL> set echo on
SQL> SELECT property_name, property_value
  2  FROM database_properties
  3  WHERE property_name LIKE 'DEFAULT_%TABLE%';

PROPERTY_NAME          PROPERTY_VALUE
-----
DEFAULT_PERMANENT_TABLESPACE SYSTEM
DEFAULT_TEMP_TABLESPACE      TEMP

SQL>

```

13. Manage default permanent and temporary tablespaces of users.

- a. Create a common user c##u in the root container.

```

SQL> CONNECT system/WElcome123##
Connected.
SQL> CREATE USER c##u IDENTIFIED BY WELCOME123##;

User created.

SQL>

```

- b. View the default tablespace and the temporary tablespace assignment for user c##u in all containers.

```

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

SQL>

```

- c. Create a local user lu in ORCLPDB1.

```
SQL> CONNECT system/WELCOME123##@orclpdb1
Connected.

SQL> CREATE USER lu IDENTIFIED BY WELCOME123##;

User created.

SQL>
```

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

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

SQL>
```

- e. Change the temporary tablespace assignment for user lu to my_temp.

```
SQL> ALTER USER lu TEMPORARY TABLESPACE my_temp;
User altered.

SQL>
```

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

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

SQL>
```

14. Log out of SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

15. Run the `reset_DBMod_Storage.sh` script to clean up the environment.

```
[oracle@edvmr1p0  
~]$ $HOME/labs/DBMod_Storage/reset_DBMod_Storage.sh  
...  
Version 23.4.0.23.11  
[oracle@edvmr1p0 ~]$
```

16. Close all terminals.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 16: Improving Space Usage

Overview

In these practices, you will use the Segment and Compression Advisors to manage space in your database. Finally, you will enable the Resumable Space Allocation feature.

Time Estimate:

It is estimated that this practice can be completed in 40 minutes.

For Instructor Use Only.
This document should not be distributed.

Practice 16-1: 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. Open a terminal window and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1  
...  
SQL>
```

2. 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 end 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>
```

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

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

```
Session altered.
```

```
SQL>
```

- b. Query the `WARNING_VALUE` and the `CRITICAL_VALUE` columns in the `DBA_THRESHOLDS` view for tablespace usage. 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>
```

4. In ORCLPDB1, create a new tablespace called `TBSALERT` with a 120MB file called `tbsalert.dbf`. Make sure that this tablespace is locally managed and uses Automatic Segment Space Management. Do not make it `auto-extensible` and do not specify any thresholds for this tablespace.

- a. Connect to ORCLPDB1.

```
SQL> ALTER SESSION SET container = orclpdb1;
```

```
Session altered.
```

```
SQL>
```

- b. Create the TBSALERT tablespace by executing the
`/home/oracle/labs/DBMod_Storage/Create_TBSALERT_TS.sql` script.

```

SQL> SET ECHO ON
SQL> @/home/oracle/labs/DBMod_Storage/Create_TBSALERT_TS.sql
SQL> CREATE TABLESPACE tbsalert
  2  DATAFILE
  ' /u01/app/oracle/oradata/ORCLCDB/orclpdb1/tbsalert.dbf'
  3  SIZE 120M REUSE LOGGING EXTENT MANAGEMENT LOCAL
  4  SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SQL>

```

5. Query how much free space the TBSALERT tablespace holds by executing the
`home/oracle/labs/DBMod_Storage/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>

```

6. Modify the threshold values for the Tablespace Space Usage metric for the TBSALERT tablespace. Set the Warning Threshold to 55 and the Critical Threshold to 70.

Note: The following command must either be all on one line or each line must end with a '-' character with no spaces following it.

```
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>
```

7. 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>
```

8. 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>
```

9. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

10. Execute the \$HOME/labs/seg_advsr_setup.sh shell script to create and populate new tables in the TBSALERT tablespace.

```
[oracle@edvmr1p0 ~]$ $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
...
[oracle@edvmr1p0 ~]$
```

11. Check the fullness level of the TBSALERT tablespace to see if the warning level has been reached.
 - a. Start SQL*Plus and connect to ORCLPDB1 as the SYSTEM user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1
...
SQL>
```

- b. Query the size of the TBSALERT tablespace. The results show that the tablespace is 60 percent full.

```
SQL> SELECT sum(bytes) * 100 / 125829120 FROM dba_extents
      WHERE tablespace_name='TBSALERT';

SUM(BYTES)*100/125829120
-----
60

SQL>
```

- c. Query the number of free bytes that are left in the TBSALERT tablespace by executing the \$HOME/labs/DBMod_Storage/TBSALERT_free_space.sql script. Recall that you created the tablespace with 120MB (125829120 bytes) of space. The query result shows that there are 125829120 bytes free and the tablespace is 39 percent free.

```

SQL> SET ECHO ON
SQL> @$HOME/labs/DBMod_Storage/TBSALERT_free_space.sql
SQL> SELECT df.tablespace_name tablespace, fs.bytes free,
   df.bytes, fs.bytes *100/ df.bytes PCT_FREE
   FROM dba_data_files df, dba_free_space fs
   WHERE df.tablespace_name = fs.tablespace_name
   AND df.tablespace_name = 'TBSALERT';

TABLESPACE          FREE      BYTES    PCT_FREE
-----  -----  -----
TBSALERT           49283072  125829120 39.1666667

SQL>

```

- d. Wait for a few minutes and 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 for 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.

12. 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>
```

13. Wait for a few minutes.

14. Query the fullness of the tablespace. The result shows that the tablespace is 75 percent full.

```
SQL> SELECT sum(bytes) * 100 / 125829120
      FROM dba_extents
     WHERE tablespace_name='TBSALERT';

SUM(BYTES)*100/125829120
-----
75

SQL>
```

15. 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 for 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>
```

16. 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>
```

17. 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:

18. Create a Segment Advisor task to get recommendations about the current space situation by executing the `home/oracle/labs/DBMod_Storage/seg_advsr_task.sql` script.

```
SQL> SET ECHO ON
SQL> @$HOME/labs/DBMod_Storage/seg_advsr_task.sql
SQL> DECLARE
  2  tname VARCHAR2(128) := 'my_seg_task';
  3  tname_desc  VARCHAR2(128) := 'Get shrink advice for segments
in TBSALERT';
  4  task_id NUMBER;
  5  object_id NUMBER;
  6  objectname VARCHAR2(100);
  7  objecttype VARCHAR2(100);
  8  BEGIN
  9  dbms_advisor.create_task('Segment Advisor',
task_id,tname,tname_desc,NULL);
 10 dbms_advisor.create_object(tname,'TABLESPACE','TBSALERT','','',
NULL,' ',object_id);
 11 dbms_advisor.set_task_parameter(tname,'RECOMMEND_ALL','TRUE');
 12 END;
 13 /
PL/SQL procedure successfully completed.

SQL>
```

19. Execute the task.

```
SQL> DECLARE
      tname VARCHAR2(128) := 'my_seg_task';
      BEGIN
      dbms_advisor.EXECUTE_TASK(tname);
      END;
      /
PL/SQL procedure successfully completed.

SQL>
```

20. 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>
```

21. Execute the \$HOME/labs/DBMod_Storage/segments_to_shrink.sql script to find out which segments should be shrunk to reclaim space. The result shows that the first three segments should be shrunk.

```
SQL> @/home/oracle/labs/DBMod_Storage/segments_to_shrink.sql
SQL> col attr1 format a5
SQL> col attr2 format a12
SQL> col message format a55
SQL> set echo on
SQL> SELECT attr1, attr2, message FROM dba_advisor_findings f,
      dba_advisor_objects o WHERE f.task_name = o.task_name AND
      f.object_id = o.object_id AND f.task_name = 'my_seg_task';

ATTR1 ATTR2          MESSAGE
----- -----
HR     EMPLOYEES2    Perform shrink, estimated savings is 18873242
bytes.
```

```

HR     EMPLOYEES1    Perform shrink, estimated savings is 18873242
bytes.
HR     EMPLOYEES3    Perform shrink, estimated savings is 18873242
bytes.
HR     EMPLOYEES4    The free space in the object is less than
10MB.
HR     EMPLOYEES5    The free space in the object is less than
10MB.

SQL>

```

22. 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>

```

23. 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 percent full from 75 percent full.

```

SQL> SELECT sum(bytes) * 100 / 125829120
      FROM dba_extents
      WHERE tablespace_name='TBSALERT';

SUM(BYTES)*100/125829120
-----
30.15625

SQL>

```

24. Drop the TBSALERT tablespace.

```
SQL> DROP TABLESPACE tbsalert INCLUDING CONTENTS AND DATAFILES;  
  
Tablespace dropped.  
  
SQL>
```

25. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

26. Close all terminals.

Practice 16-2: Using Compression

Overview

In this practice, you will use Advanced Index Compression to reduce the storage for indexes. You will 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 execute the `$HOME/labs/DBMod_Storage/setup_index.sh` shell script to create an index on the `HR.TEST` table in `ORCLPDB1`.

```
[oracle@edvmr1p0 ~]$ $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
...
[oracle@edvmr1p0 ~]$
```

2. Start SQL*Plus and connect to ORCLPDB1 as the HR user.

```
[oracle@edvmr1p0 ~]$ sqlplus hr/WELCOME123##@orclpdb1  
...  
SQL>
```

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

```
SQL> COL index_name FORMAT A20  
SQL> SELECT index_name, compression FROM user_indexes WHERE  
index_name = 'I_TEST';  
  
INDEX_NAME          COMPRESSION  
-----  
I_TEST              DISABLED  
  
SQL>
```

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

```
SQL> SELECT blocks FROM user_segments WHERE  
segment_name='I_TEST';  
  
BLOCKS  
-----  
1152  
  
SQL>
```

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

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

6. 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.

```
[oracle@edvmr1p0 ~]$ 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;
...
[oracle@edvmr1p0 ~]$
```

- Start SQL*Plus and connect to ORCLPDB1 as the hr user.

Hint: Use the up arrow key several times to recall the command from the OS command-line buffer.

```
[oracle@edvmr1p0 ~]$ sqlplus hr/WELCOME123##@orclpdb1
...
SQL>
```

- 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_ncmp => row_ncmp,
cmp_ratio => cmp_ratio,
comptype_str => comptype_str,
subset_numrows => dbms_compression.COMP_RATIO_MINROWS,
objtype => dbms_compression.OBJTYPE_INDEX
);
DBMS_OUTPUT.PUT_LINE('Block used by compressed index = ' ||
blkcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Block used by uncompressed index = ' ||
blkcnt_ncmp);
DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
DBMS_OUTPUT.PUT_LINE('Compression ratio org = ' || cmp_ratio);
END;
/

```

Block used by compressed index = 809
 Block used by uncompressed index = 1029
 Compression type = "Compress Advanced Low"
 Compression ratio org = 1

PL/SQL procedure successfully completed.

SQL>

9. Use the Compression Advisor again to get recommendations about the space you would save by compressing the index with the COMP_INDEX_ADVANCED_HIGH compression level by executing the \$HOME/labs/DBMod_Storage/Compression_index_high.sql script. The result indicates that the space used by the index would be reduced down to 130 blocks. The Advanced High Compression ratio is equal to 8.

```

SQL> @$HOME/labs/DBMod_Storage/Compression_index_high.sql
SQL> set serveroutput on
SQL> DECLARE
blkcnt_cmp pls_integer;
blkcnt_ncmp pls_integer;
row_cmp pls_integer;
row_ncmp pls_integer;
cmp_ratio pls_integer;
comptype_str varchar2(100);
BEGIN
DBMS_COMPRESSION.GET_COMPRESSION_RATIO
(
scratchtbsname => 'USERS',

```

```

ownname => 'HR',
objname => 'I_TEST',
subobjname => NULL,
comptype => dbms_compression.COMP_INDEX_ADVANCED_HIGH,
blkcnt_cmp => blkcnt_cmp,
blkcnt_ncmp => blkcnt_ncmp,
row_cmp => row_cmp,
row_ncmp => row_ncmp,
cmp_ratio => cmp_ratio,
comptype_str => comptype_str,
subset_numrows => dbms_compression.COMP_RATIO_MINROWS,
objtype => dbms_compression.OBJTYPE_INDEX
);
DBMS_OUTPUT.PUT_LINE('Block used by compressed index = ' || blkcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Block used by uncompressed index = ' || blkcnt_ncmp);
DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
DBMS_OUTPUT.PUT_LINE('Compression ratio org = ' || cmp_ratio);
END;
/
Block used by compressed index = 130
Block used by uncompressed index = 1029
Compression type = "Compress Advanced High"
Compression ratio org = 8
PL/SQL procedure successfully completed.

SQL>
```

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

10. Rebuild the index with Advanced High Compression.

```

SQL> ALTER INDEX hr.i_test REBUILD COMPRESS ADVANCED HIGH;

Index altered.

SQL>
```

11. 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>
```

12. Query the space used by the index created on the HR.TEST table. The space now has 256 blocks.

```
SQL> SELECT blocks FROM user_segments WHERE
segment_name='I_TEST';

BLOCKS
-----
256

SQL>
```

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

Answer: Yes.

13. Revert back to the initial compression level.

```
SQL> ALTER INDEX hr.i_test REBUILD NOCOMPRESS;

Index altered.

SQL>
```

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

```
SQL> SELECT blocks FROM user_segments WHERE
segment_name='I_TEST';

BLOCKS
-----
1152

SQL>
```

15. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

For Instructor Use Only.
This document should not be distributed.

Practice 16-3: Enabling the Resumable Space Allocation Feature

Overview

In this practice, you will 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), and space quota exceeded condition (ORA-01536)

Tip

Because you will 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. Open a terminal window; start SQL*Plus and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WElcome123##@orclpdb1
...
SQL>
```

2. Execute the \$HOME/labs/DBMod_Storage/CreateINVENTORYTablespace.sql script to create an unpopulated tablespace named INVENTORY.

```
SQL> SET ECHO ON
SQL> @$HOME/labs/DBMod_Storage/CreateINVENTORYTablespace.sql
SQL> DROP TABLESPACE INVENTORY INCLUDING CONTENTS AND DATAFILES;
SQL> DROP TABLESPACE INVENTORY INCLUDING CONTENTS AND DATAFILES
```

```
*  
ERROR at line 1:  
ORA-00959: tablespace 'INVENTORY' does not exist  
Help: https://docs.oracle.com/error-help/db/ora-00959/  
  
SQL>  
SQL> CREATE SMALLFILE TABLESPACE INVENTORY  
2      DATAFILE  
3  
' /u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF' SIZE  
5242880  
4      DEFAULT NOCOMPRESS  
5      ONLINE  
6      SEGMENT SPACE MANAGEMENT AUTO  
7      EXTENT MANAGEMENT LOCAL AUTOALLOCATE;  
  
Tablespace created.  
  
SQL>
```

3. Execute the \$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. Partway 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
...
[oracle@edvmr1p0 ~] $
```

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.

- Question:** Are the rows that were inserted into the table lost or definitely inserted?
Answer: 2048 rows were inserted.
- 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.

- Start SQL*Plus again and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~] $ sqlplus system/WElcome123##@orclpdb1
...
SQL>
```

- Enable the resumable mode.

```
SQL> ALTER SESSION enable resumable;

Session altered.

SQL>
```

6. Re-execute the CreateTable_X script. The script will suspend this time.

```
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 ;
```

Question: Why is the script suspended?

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

Question: Is there any warning message to tell you the load is suspended?

Answer: No. If the script does not execute any further, you must 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:

7. Open another terminal window. This will be referred to as Window 2.
8. Start SQL*Plus and connect to ORCLPDB1 as system.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1
...
SQL>
```

- 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> COL sql_text FORMAT A60
SQL> COL 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 increase tablespace INVENTORY by 1MB during
insert or update on table SYSTEM.X

SQL>
```

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

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ tail -30
/u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace/alert_orclcdb.lo
g
...
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
[oracle@edvmr1p0 ~]$
```

12. Proceed with the appropriate corrective action. Because the INVENTORY tablespace is not autoextensible, you can configure it as autoextensible with a size limit.
 - a. Start SQL*Plus and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WElcome123##@orclpdb1
...
SQL>
```

- b. Query the DBA_DATA_FILES view to verify whether the INVENTORY tablespace is autoextensible. The result shows that the tablespace is not.

```
SQL> COL file_name FORMAT A60
SQL> SELECT file_name, autoextensible FROM dba_data_files
WHERE tablespace_name='INVENTORY';
```

FILE_NAME	AUT
/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF	NO

```
SQL>
```

- c. Enable autoextend for the INVENTORY01.DBF data file.

```
SQL> ALTER DATABASE DATAFILE
'/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF'
autoextend ON maxsize 10m;
```

```
Database altered.
```

```
SQL>
```

- d. Query the DBA_DATA_FILES view again to verify whether the INVENTORY tablespace is autoextensible. The result shows that it is.

```
SQL> SELECT file_name, autoextensible FROM dba_data_files
WHERE tablespace_name='INVENTORY';
```

FILE_NAME	AUT
/u01/app/oracle/oradata/ORCLCDB/orclpdb1/INVENTORY01.DBF	YES

```
SQL>
```

Window 1: Check the Suspended Session:

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

```
SQL> INSERT INTO x
  2  SELECT * FROM x ;

2048 rows created.

SQL> COMMIT;

Commit complete.

SQL> quit
...
[oracle@edvmr1p0 ~] $
```

14. Close the terminal window.

Window 2: Verify That There Are No Suspended Sessions:

15. 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>
```

16. Exit from SQL*Plus and close the terminal window.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~] $
```

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 17: Managing Undo Data

Overview

In these practices, you will view the undo configuration of the database.

Time Estimate

It is estimated that this practice will complete in 2 minutes.

Practice 17-1: Viewing Undo Tablespaces in a CDB

Overview

In this practice, you will view undo tablespaces and validate the local management configuration of the CDB and PDBs.

Tasks

1. Start SQL*Plus and connect to the CDB\$ROOT as system.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##  
...  
SQL>
```

2. Display the undo tablespaces used in the CDB.

Note: Your output may be different.

```
SQL> SELECT file#, ts.name, ts.ts#, ts.con_id  
      FROM v$logfile d, v$tablespace ts  
     WHERE d.ts#=ts.ts#  
       AND d.con_id=ts.con_id  
       AND ts.name like 'UNDO%';  
  
FILE# NAME          TS# CON_ID  
----- -----  
      4 UNDOTBS1        2    1  
      8 UNDOTBS1        2    2  
     11 UNDOTBS1        2    3  
     15 UNDOTBS1        2    4  
     49 UNDOTBS1        2    5  
  
SQL>
```

Question: According to the list of undo tablespaces, what can you conclude about the undo mode used?

Answer: Because there is an undo tablespace in each container, the undo mode used is the local undo mode.

Question: Why should you use the local undo mode?

Answer: The local undo mode is useful for hot cloning, PDB relocation, and PDB proxying.

3. Verify that the undo mode is LOCAL.

```
SQL> SELECT property_name, property_value  
      FROM database_properties  
     WHERE property_name = 'LOCAL_UNDO_ENABLED';  
  
PROPERTY_NAME          PROPERTY_VALUE  
-----  
LOCAL_UNDO_ENABLED        TRUE  
  
SQL>
```

4. Exit from SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

5. Close all open terminals.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 18: Creating and Managing User Accounts

Overview

In these practices, you will examine OS users and groups associated to Oracle Database, create database users, and grant roles and privileges to users.

For Instructor Use Only.
This document should not be distributed.

Practice 18-1: Creating Common and Local Users

Overview

In this practice, you will log in 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, grant this user the `DBA` role also and the `CREATE SESSION` privilege in all the 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 the `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 a 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. In addition, 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 the Common Users:

1. Open terminal and execute

```
/home/oracle/labs/DBMod_UsersSec/reset_ORCLPDB1.sh.
```

Note: It may take a few minutes to complete; please wait for the activity to complete and for the prompt to return before continuing with the next step. Error messages saying “the object does not exist” can be ignored.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_UsersSec/reset_ORCLPDB1.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. Reset the common users.

Note: Error messages saying “the object does not exist” can be ignored.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_UsersSec/reset_users_roles.sh  
...  
[oracle@edvmr1p0 ~]$
```

Create c##CDB_ADMIN1:

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

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

4. Create a common user named `c##cdb_admin1` by using the `CREATE USER` command. Set the `USERS` tablespace as the default and `TEMP` as the temporary tablespace. Also, unlock the account so that `C##CDB_ADMIN1` can log in right away.

Important! To create a common user, you must start the user name with `C##` or `C##`, and you may 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.

```
SQL> CREATE USER c##cdb_admin1 IDENTIFIED BY WELCOME123##  
      container=all DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp  
      ACCOUNT UNLOCK;
```

User created.

```
SQL>
```

5. Grant `C##CDB_ADMIN1` the `DBA` role, the `CREATE SESSION` privilege, and the `SYSDBA` privilege in all containers. This is an example of granting privileges and a role commonly.

```
SQL> GRANT create session, dba, sysdba TO c##cdb_admin1  
      container=all;
```

Grant succeeded.

```
SQL>
```

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

```
GRANT create session, dba, sysdba 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.

6. List the common users by querying the DBA_USERS view. Scroll down and verify that c##cdb_admin1 is included.

```
SQL> COL username FORMAT A20
SQL> SELECT DISTINCT username FROM dba_users WHERE common='YES'
      ORDER BY username;

USERNAME
-----
ANONYMOUS
APPQOSSYS
AUDSYS
C##CDB_ADMIN1
CTXSYS
...
SYS
SYS$UMF
SYSBACKUP
SYSDG
SYSKM
SYSRAC
SYSTEM
WMSYS
XDB
XS$NULL

34 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.

7. Disconnect from the root container.

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

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

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

9. 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, who is a member of the DBA group, you can enter anything as a username and anything as a password and be connected as username and passwords entered are ignored when the service name is not specified and mapped from OS session.

```
SQL> CONNECT anyuser/anystring@ORCLCDB AS SYSDBA
ERROR:
ORA-01017: invalid credential or not authorized; logon denied
Help: https://docs.oracle.com/error-help/db/ora-01017/
```

Warning: You are no longer connected to ORACLE.

```
SQL> CONNECT anyuser/anystring AS sysdba
Connected.
SQL>
```

10. Show the current container name.

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

11. Show the current user. The current user is `sys`, which means the `snyuser` 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>
```

12. View the list of privileges for the session 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  
  
274 rows selected.  
  
SQL>
```

13. Disconnect from the root container.

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

14. Connect to the root container as c##cdb_admin1, but do not exercise the SYSDBA privilege.

```
SQL> CONNECT c##cdb_admin1/WELCOME123##  
Connected.  
SQL>
```

15. 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 the 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>
```

16. View the list of privileges for the c##xcdb_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 the user was connected with the SYSDBA privilege.

```
SQL> SELECT * FROM session_privs ORDER BY privilege;

PRIVILEGE
-----
ADMINISTER ANY SQL TUNING SET
ADMINISTER DATABASE TRIGGER
...
UPDATE ANY TABLE
USE ANY JOB RESOURCE
USE ANY SQL TRANSLATION PROFILE

254 rows selected.

SQL>
```

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

```
SQL> ALTER SESSION SET container = orclpdb1;

Session altered.

SQL>
```

18. Show the current container. It is ORCLPDB1.

```
SQL> SHOW con_name

CON_NAME
-----
ORCLPDB1
SQL>
```

Create the PDB1_ADMIN User:

You had just connected to ORCLPDB1 as c##cdb_admin1.

Note: You need to be logged in to ORCLPDB1 to create a local administrator for ORCLPDB1. The c##cdb_admin1 user can create the pdb1_admin user.

19. 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 the user can log in right away. Because this is a local user and not a common user, do not include the CONTAINER=ALL clause.

```
SQL> CREATE USER pdb1_admin IDENTIFIED BY WElcome123## DEFAULT  
TABLESPACE users TEMPORARY TABLESPACE temp ACCOUNT unlock;  
  
User created.  
  
SQL>
```

20. Grant pdb_admin1 the DBA role and the CREATE SESSION privilege in ORCLPDB1 only.

Note: This is an example of granting a privilege and role locally. Since you are connected to the PDB, the container=local is the default.

```
SQL> GRANT create session, dba TO pdb1_admin;  
  
Grant succeeded.  
  
SQL>
```

21. List the local user accounts for ORCLPDB1 by querying the DBA_USERS view. The pdb_admin1 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>
```

22. Disconnect c##CDB_ADMIN1 from PDB1.

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

23. Connect to ORCLPDB1 as PDB1_ADMIN.

```
SQL> CONNECT pdb1_admin/WELCOME123##@orclpdb1
Connected.
SQL>
```

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

```
SQL> SHOW USER
USER is "PDB1_ADMIN"
SQL>
```

25. View the list of privileges for `pdb_admin1` by querying the `SESSION_PRIVS` view.

```
SQL> SELECT * FROM session_privs ORDER BY privilege;

PRIVILEGE
-----
ADMINISTER ANY SQL TUNING SET
ADMINISTER DATABASE TRIGGER
...
UPDATE ANY TABLE
USE ANY JOB RESOURCE
USE ANY SQL TRANSLATION PROFILE

254 rows selected.

SQL>
```

26. Try to connect to the root container as the `pdb_admin1` 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 the `CREATE SESSION` privileges in all containers, including the root container. `pdb_admin1` has the same role and privilege, but only in `ORCLPDB1`.

```
SQL> ALTER SESSION SET container = cdb$root;
ERROR:
ORA-01031: insufficient privileges
Help: https://docs.oracle.com/error-help/db/ora-01031/
SQL>
```

27. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

28. Exit all terminals.

Practice 18-2: Creating a Local User for an Application

Overview

In this practice, you will 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, and a local user account called `appuser`, which is used by the application for querying the data. `INVENTORY` and `appuser` are examples 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. **(Catchup Step)** If you *skipped Practice 18-1*, open a terminal window and run the `/home/oracle/labs/DBMod_UsersSec/setup_users_lab18.sh` script to catch up. Otherwise, proceed to **step 2**.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_UsersSec/setup_users_lab18.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. Open a terminal and start SQL*Plus and connect to ORCLPDB1 as the `PDB1_ADMIN` user.

```
[oracle@edvmr1p0 ~]$ sqlplus pdb1_admin/WELCOME123##@orclpdb1  
...  
SQL>
```

3. Create a local user account named `INVENTORY`. Set the default tablespace to the `USERS` tablespace and grant unlimited quota on that tablespace.

```
SQL> CREATE USER inventory IDENTIFIED BY WELCOME123##  
      DEFAULT TABLESPACE users QUOTA UNLIMITED ON users;  
  
User created.  
  
SQL>
```

4. Grant the CREATE SESSION privilege and db_developer_role role to the INVENTORY user.

Note: The db_developer_role is a new role created in Oracle Database 23c, which includes all privileges needed to manage a complex application schema.

```
SQL> GRANT create session, db_developer_role TO inventory;  
  
Grant succeeded.  
  
SQL>
```

5. 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 the Privileges:

6. Disconnect from the connection as pdb1_admin from ORCLPDB1.

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

7. Verify that the INVENTORY user account can connect to ORCLPDB1.

```
SQL> CONNECT inventory/WElcome123##@orclpdb1
Connected.
SQL>
```

8. List the privileges for INVENTORY by querying the SESSION_PRIVS view. The results show that INVENTORY user has many privileges including the CREATE SESSION privilege.

```
SQL> SELECT * FROM session_privs ORDER BY 1;
```

```
PRIVILEGE
```

```
-----  
CREATE ANALYTIC VIEW  
CREATE ATTRIBUTE DIMENSION  
CREATE CLUSTER  
CREATE CUBE  
CREATE CUBE BUILD PROCESS  
CREATE CUBE DIMENSION  
CREATE DIMENSION  
CREATE DOMAIN  
CREATE HIERARCHY  
CREATE INDEXTYPE  
CREATE JOB  
CREATE MATERIALIZED VIEW  
CREATE MINING MODEL  
CREATE MLE  
CREATE OPERATOR  
CREATE PROCEDURE  
CREATE PROPERTY GRAPH  
CREATE SEQUENCE  
CREATE SESSION  
CREATE SYNONYM  
CREATE TABLE  
CREATE TRIGGER  
CREATE TYPE  
CREATE VIEW  
DEBUG CONNECT SESSION  
EXECUTE DYNAMIC MLE  
FORCE TRANSACTION  
ON COMMIT REFRESHCREATE SESSION
```

```
28 rows selected.  
SQL>
```

Note: The above query only includes a partial set of privileges that are included in the db_developer_role included in the session_privs view.

9. While connected to the orclpdb1 as the inventory user, create a table called table1 and insert a row in it then commit.

```
SQL> CREATE TABLE table1 (id number, value char(1));  
  
Table created.  
  
SQL> INSERT INTO table1 VALUES (1, 'A');  
  
1 row created.  
  
SQL> COMMIT.  
  
Commit complete.  
SQL>
```

10. Connect to the orclpdb1 PDB as pdb1_admin, execute the script file /home/oracle/labs/DBMod_UsersSec/devroleprivs.sql, and examine the full list of privileges included in the db_developer_role.

Note: The following procedure is used to display privileges and roles included in db_developer_role:

```
DECLARE  
procedure printRolePrivileges(p_role in varchar2,  
p_spaces_to_indent in number)  
IS v_child_roles DBMS_SQL.VARCHAR2_TABLE;  
v_system_privs DBMS_SQL.VARCHAR2_TABLE;  
v_table_privs DBMS_SQL.VARCHAR2_TABLE;  
v_indent_spaces varchar2(2048);  
BEGIN  
for space in 1..p_spaces_to_indent LOOP  
v_indent_spaces := v_indent_spaces || ' ';  
end LOOP;
```

```

select PRIVILEGE bulk collect into v_system_privs from
DBA_SYS_PRIVS where GRANTEE = p_role order by PRIVILEGE;
for privind in 1..v_system_privs.COUNT LOOP
DBMS_OUTPUT.PUT_LINE(v_indent_spaces || 'System priv: ' ||
v_system_privs(privind));
END LOOP;
select PRIVILEGE || ' ' || OWNER || '.' || TABLE_NAME bulk
collect into v_table_privs from DBA_TAB_PRIVS where GRANTEE =
p_role order by TABLE_NAME asc;
for tabprivind in 1..v_table_privs.COUNT LOOP
DBMS_OUTPUT.PUT_LINE(v_indent_spaces || 'Object priv: ' ||
v_table_privs(tabprivind));
END LOOP;
select GRANTED_ROLE bulk collect into v_child_roles from
DBA_ROLE_PRIVS where GRANTEE= p_role order by GRANTED_ROLE asc;
for roleind in 1..v_child_roles.COUNT LOOP
DBMS_OUTPUT.PUT_LINE(v_indent_spaces || 'Role priv: ' ||
v_child_roles(roleind));
printRolePrivileges(v_child_roles(roleind), p_spaces_to_indent +
2);
END LOOP;
EXCEPTION when OTHERS then DBMS_OUTPUT.PUT_LINE('Got exception: ' ||
SQLERRM );
END printRolePrivileges;
BEGIN printRolePrivileges('DB_DEVELOPER_ROLE', 0);
END;
/

```

```

SQL> CONN pdb1_admin/WELCOME123##@orclpdb1
Connected.

SQL> @/home/oracle/labs/DBMod_UsersSec/devroleprivs.sql
SQL> set serveroutput on format wrapped
SQL> DECLARE
 2  procedure printRolePrivileges(p_role in varchar2,
p_spaces_to_indent in number)
 3  IS v_child_roles DBMS_SQL.VARCHAR2_TABLE;
 4  v_system_privs DBMS_SQL.VARCHAR2_TABLE;
 5  v_table_privs DBMS_SQL.VARCHAR2_TABLE;
 6  v_indent_spaces varchar2(2048);

```

```

7  BEGIN
8  for space in 1..p_spaces_to_indent LOOP
9    v_indent_spaces := v_indent_spaces || ' ';
10   end LOOP;
11  select PRIVILEGE bulk collect into v_system_privs from
DBA_SYS_PRIVS where GRANTEE = p_role order by PRIVILEGE;
12  for privind in 1..v_system_privs.COUNT LOOP
DBMS_OUTPUT.PUT_LINE(v_indent_spaces || 'System priv: ' ||
v_system_privs(privind));
13  END LOOP;
14  select PRIVILEGE || ' ' || OWNER || '.' || TABLE_NAME bulk
collect into v_table_privs from DBA_TAB_PRIVS where GRANTEE =
p_role order by TABLE_NAME asc;
15  for tabprivind in 1..v_table_privs.COUNT LOOP
DBMS_OUTPUT.PUT_LINE(v_indent_spaces || 'Object priv: ' ||
v_table_privs(tabprivind));
16  END LOOP;
17  select GRANTED_ROLE bulk collect into v_child_roles from
DBA_ROLE_PRIVS where GRANTEE= p_role order by GRANTED_ROLE asc;
18  for roleind in 1..v_child_roles.COUNT LOOP
DBMS_OUTPUT.PUT_LINE(v_indent_spaces || 'Role priv: ' ||
v_child_roles(roleind));
19  printRolePrivileges(v_child_roles(roleind),
p_spaces_to_indent + 2);
20  END LOOP;
21  EXCEPTION when OTHERS then DBMS_OUTPUT.PUT_LINE('Got
exception: ' || SQLERRM );
22  END printRolePrivileges;
23  BEGIN printRolePrivileges('DB_DEVELOPER_ROLE', 0);
24  END;
25  /
System priv: CREATE CUBE
System priv: CREATE CUBE BUILD PROCESS
System priv: CREATE CUBE DIMENSION
System priv: CREATE DIMENSION
System priv: CREATE DOMAIN
System priv: CREATE JOB
System priv: CREATE MINING MODEL
System priv: CREATE MLE
System priv: CREATE SESSION
System priv: DEBUG CONNECT SESSION
System priv: EXECUTE DYNAMIC MLE
System priv: FORCE TRANSACTION
System priv: ON COMMIT REFRESH
Object priv: SELECT SYS.DBA_PENDING_TRANSACTIONS

```

```
Object priv: EXECUTE SYS.DBMS_REDACT
Object priv: EXECUTE SYS.DBMS_RLS
Object priv: EXECUTE SYS.DBMS_TSDP_MANAGE
Object priv: EXECUTE SYS.DBMS_TSDP_PROTECT
Object priv: EXECUTE SYS.JAVASCRIPT
Object priv: READ SYS.V_$PARAMETER
Object priv: READ SYS.V_$STATNAME
Role priv: CTXAPP
    System priv: CREATE SEQUENCE
    Object priv: EXECUTE CTXSYS.CTX_ANL
    Object priv: EXECUTE CTXSYS.CTX_DDL
    Object priv: EXECUTE CTXSYS.CTX_ENTITY
    Object priv: EXECUTE CTXSYS.CTX_OUTPUT
    Object priv: EXECUTE CTXSYS.CTX_THES
    Object priv: EXECUTE CTXSYS.CTX_ULEXER
    Object priv: INSERT CTXSYS.DR$DICTIONARY
    Object priv: DELETE CTXSYS.DR$DICTIONARY
    Object priv: SELECT CTXSYS.DR$DICTIONARY
    Object priv: UPDATE CTXSYS.DR$DICTIONARY
    Object priv: INSERT CTXSYS.DR$THS
    Object priv: INSERT CTXSYS.DR$THS_BT
    Object priv: INSERT CTXSYS.DR$THS_FPHRASE
    Object priv: UPDATE CTXSYS.DR$THS_PHRASE
    Object priv: INSERT CTXSYS.DR$THS_PHRASE
    Object priv: EXECUTE CTXSYS.DRIENTL
    Object priv: EXECUTE CTXSYS.DRITHSL
Role priv: RESOURCE
    System priv: CREATE ANALYTIC VIEW
    System priv: CREATE ATTRIBUTE DIMENSION
    System priv: CREATE CLUSTER
    System priv: CREATE HIERARCHY
    System priv: CREATE INDEXTYPE
    System priv: CREATE MATERIALIZED VIEW
    System priv: CREATE OPERATOR
    System priv: CREATE PROCEDURE
    System priv: CREATE PROPERTY GRAPH
    System priv: CREATE SEQUENCE
    System priv: CREATE SYNONYM
    System priv: CREATE TABLE
    System priv: CREATE TRIGGER
    System priv: CREATE TYPE
    System priv: CREATE VIEW
```

```

Role priv: SODA_APP
Object priv: EXECUTE XDB.DBMS_SODA_ADMIN
Object priv: EXECUTE XDB.DBMS_SODA_USER_ADMIN
Object priv: READ XDB.JSON$USER_COLLECTION_METADATA

PL/SQL procedure successfully completed.

SQL>

```

Use Schema Level Privileges to Manage Application Users:

11. Connected to `orclpdb1` as `pdb1_admin`, create an application user `appuser` and grant `create session` and `select any table` on schema `inventory` to the user, and then validate that the `hr.departments` table exists.

```

SQL> CONNECT pdb1_admin/WELCOME123##@orclpdb1
Connected
SQL> CREATE USER appuser IDENTIFIED BY "WELCOME123##";

User created.

SQL> GRANT create session TO appuser;

Grant succeeded.

SQL> GRANT select any table on schema inventory TO appuser;

Grant succeeded.

SQL> COL grantee FORMAT A10
SQL> COL privilege FORMAT A20
SQL> COL schema FORMAT A10
SQL> SELECT * FROM dba_schema_privs;

GRANTEE      PRIVILEGE          SCHEMA      ADM COM INH
-----      -----
APPUSER      SELECT ANY TABLE    INVENTORY   NO  NO  NO

```

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

  COUNT(*)
-----
  27

SQL>
```

12. Connect to the ORCLPDB1 PDB as an inventory user, create a table `foo`, and insert a row and then commit.

```
SQL> CONN inventory/WElcome123##@orclpdb1
Connected.

SQL> CREATE TABLE foo (id number, value char(1));

Table created

SQL> INSERT INTO foo VALUES (1, 'A');

1 row created

SQL> COMMIT;

Commit complete.

SQL>
```

13. Examine the Schema Level Privilege given to appuser by querying tables in inventory and hr schemas.

- a. Connect to ORCLPDB1 as appuser and query the `inventory.table1` table, which existed before the appuser was created, and the `inventory.foo` table that was created after the appuser was created and granted the Schema Level Privilege.

```
SQL> CONNECT appuser/WElcome123##@ORCLpdb1
Connected.

SQL> SELECT * FROM inventory.table1;

  ID V
-----
  - -
  1 A
```

```
SQL> SELECT * FROM inventory.foo;

      ID      V
      -----
        1      A

SQL>
```

- While connected to the ORCLPDB1 PDB as appuser, try to query the HR.DEPARTMENTS table.

Note: Schema level privileges were granted only on `inventory` schema and it does not expand to `hr` schema.

```
SQL> SELECT * FROM hr.departments;
SELECT * FROM sh.times
*
ERROR at line 1:
ORA-00942: table or view does not exist
Help: https://docs.oracle.com/error-help/db/ora-00942/

SQL>
```

- Connect to ORCLPDB1 as the `pdb1_admin` user, drop the user's `appuser` and `inventory`, including their objects, and the `hr.foo` table that was created for this exercise, and then exit.

```
SQL> CONN pdb1_admin/WElcome123##@ORCLpdb1
Connected.

SQL> DROP USER appuser;

User dropped.

SQL> DROP USER inventory CASCADE;

Table dropped

SQL> EXIT
```

- Close the terminal window.

Practice 18-3: Exploring OS and Password File Authentication

Overview

In this practice, you will 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 had logged in to the server as the `oracle` OS user and then to the Oracle Database as the `sys` 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. **(Catchup Step)** If you *skipped Practice 18-1*, open a terminal window and run the `/home/oracle/labs/DBMod_UsersSec/setup_users_lab18.sh` script to catch up. Otherwise, proceed to **step 2**.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_UsersSec/setup_users_lab18.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. Open a new terminal and 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.

```
[oracle@edvmr1p0 ~]$ cat /etc/group  
root:x:0:  
bin:x:1:  
daemon:x:2:  
...  
oinstall:x:54321:  
dba:x:54322:oracle  
...  
[oracle@edvmr1p0 ~]$
```

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

```
[oracle@edvmr1p0 ~]$ whoami  
oracle  
[oracle@edvmr1p0 ~]$
```

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

- a. The `/etc/passwd` file is a text file that lists user account information needed for logging in to the OS. Execute the following command to search for the `oracle` user. The format of the row is `user:password:user ID:primary group ID:home directory:login_shell`. Passwords are stored in the `/etc/shadow` file, so an `x` is used here as a placeholder.

```
[oracle@edvmr1p0 ~]$ grep oracle /etc/passwd  
oracle:x:1000:54321::/home/oracle:/bin/bash  
[oracle@edvmr1p0 ~]$
```

- b. The above information tells you that `oracle`'s primary group ID is 1000. 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.

```
[oracle@edvmr1p0 ~]$ grep oinstall /etc/group  
oinstall:x:54321:  
[oracle@edvmr1p0 ~]$
```

- c. Investigate further. Search for `oracle` in the `group` file. The results tell you that `oracle` is a user in the `dba` group. The `dba` group is the Database Administrator Group, and any user in this group has the `SYSDBA` system privilege. So, if you log in 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 in using OS authentication, all you need to specify is `CONNECT / AS SYSDBA`. The `/` tells SQL*Plus to look up the privileges for the OS user's group.

```
[oracle@edvmr1p0 ~]$ grep oracle /etc/group  
oracle:x:1000:  
dba:x:54322:oracle  
oper:x:54323:oracle  
fuse:x:54331:oracle  
[oracle@edvmr1p0 ~]$
```

- d. An alternate way to get all of this information for the current user in a single command is the command `id`.

```
[oracle@edvmr1p0 ~]$ id  
uid=1000(oracle) gid=54321(oinstall)  
groups=54321(oinstall),54322(dba),54323(oper),54331(fuse)  
[oracle@edvmr1p0 ~]$
```

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.

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

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

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

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

Name	Null?	Type
USERNAME		VARCHAR2 (128)
SYSDBA		VARCHAR2 (5)
SYSOPER		VARCHAR2 (5)
SYSASM		VARCHAR2 (5)
SYSBACKUP		VARCHAR2 (5)
SYSDG		VARCHAR2 (5)
SYSKM		VARCHAR2 (5)
ACCOUNT_STATUS		VARCHAR2 (30)
PASSWORD_PROFILE		VARCHAR2 (128)
LAST_LOGIN		TIMESTAMP (9) WITH TIME ZONE
LOCK_DATE		DATE
EXPIRY_DATE		DATE
EXTERNAL_NAME		VARCHAR2 (1024)
AUTHENTICATION_TYPE		VARCHAR2 (8)

COMMON	VARCHAR2 (3)
PASSWORD_VERSIONS	VARCHAR2 (12)
CON_ID	NUMBER
SQL>	

7. List the users in the password file by querying the V\$PFILE_USERS view.

```
SQL> COL username FORMAT A20
SQL> SELECT username FROM v$pfile_users;

USERNAME
-----
SYS
C##CDB_ADMIN1

SQL>
```

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

```
SQL> SELECT account_status, sysdba FROM v$pfile_users WHERE
username='SYS';

ACCOUNT_STATUS          SYSDBA
-----
OPEN                   TRUE

SQL>
```

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

```
SQL> EXIT
...
[oracle@edvmr1p0 ~] $
```

10. Close all terminals.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 19: Configuring Privilege and Role Authorization

Overview

In these practices, you will create roles. You will grant privileges to roles.

For Instructor Use Only.
This document should not be distributed.

Practice 19-1: Granting a Local Role (DBA) to PDBADMIN

Overview

In this practice, you will 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 more power to PDBADMIN with the DBA role so that in later practices PDBADMIN is able to create profiles, roles, and users.

Tasks

Explore the Privileges and Roles Granted to PDBADMIN

1. Open a new terminal window; start SQL*Plus and connect as the sys user with the SYSDBA privilege.

```
[oracle@edvmr1p0~ ]$ sqlplus / as sysdba  
...  
SQL>
```

2. List the system privileges granted to the pdbadmin user (this admin user was created during the PDB creation) by querying the DBA_SYS_PRIVS view. This view describes system privileges granted to users and roles. The results show that pdbadmin has been granted the UNLIMITED TABLESPACE privilege 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' ;  
  
GRANTEE    PRIVILEGE          ADM  COM  INH  
-----  -----  
PDBADMIN  UNLIMITED TABLESPACE      NO   NO   NO  
  
SQL>
```

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

```

SQL> COL granted_role FORMAT A10
SQL> SELECT granted_role, admin_option FROM cdb_role_privs WHERE
grantee='PDBADMIN';

GRANTED_ROLE ADM
-----
PDB_DBA      YES
DBA          NO

SQL>

```

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

Note: 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 information of all roles. The results show that the PDB_DBA role consists of two system privileges: CREATE SESSION and CREATE PLUGGABLE DATABASE.

```

SQL> SELECT privilege FROM role_sys_privs WHERE role='PDB_DBA'
ORDER BY 1;

PRIVILEGE
-----
CREATE PLUGGABLE DATABASE
CREATE SESSION

SQL>

```

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

```

SQL> COL granted_role FORMAT A15
SQL> SELECT granted_role FROM dba_role_privs WHERE grantee =
'PDB_DBA' ORDER BY 1;

GRANTED_ROLE
-----
CONNECT
WM_ADMIN_ROLE

SQL>

```

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

```
SQL> SELECT privilege FROM role_sys_privs WHERE role='CONNECT'  
ORDER BY 1;
```

```
PRIVILEGE  
-----  
CREATE SESSION  
SET CONTAINER  
  
SQL>
```

7. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~ ] $
```

8. Close all terminals.

Practice 19-2: Using SQL*Developer to Create Local Roles

Overview

In this practice, you will 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.

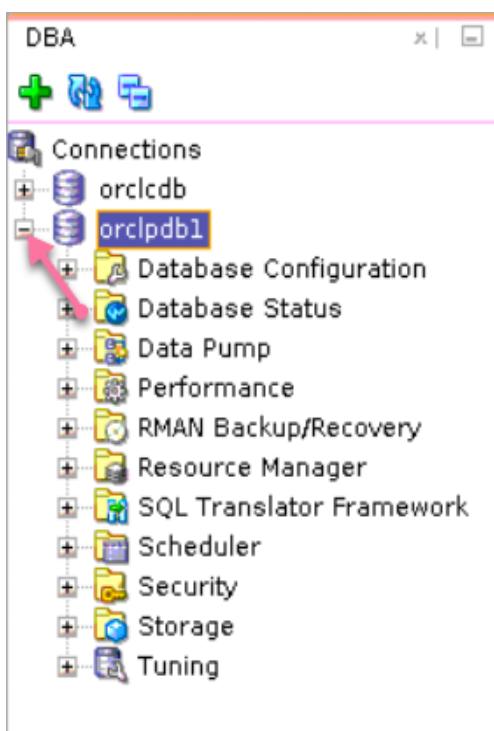
Tasks

Log in to SQL*Developer (ORCLPDB1)

1. Create a connection to ORCLPDB1 as the `sys` user if the connection does not exist.
 - a. Start SQL*Developer. The SQL*Developer icon is on the desktop.
 - b. In the Connections pane on the left, click the Name: `orclpdb1` (a connection for ORCLPDB1 as the `sys` user).
 - c. In the DBA connection box, click the connection: `orclpdb1`.

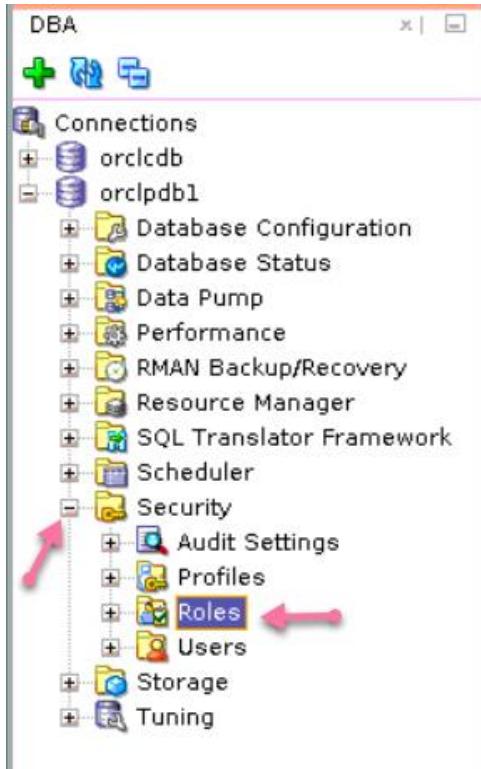
Create the HRCLERK Role

2. Expand the `orclpdb1` connection in the DBA box.



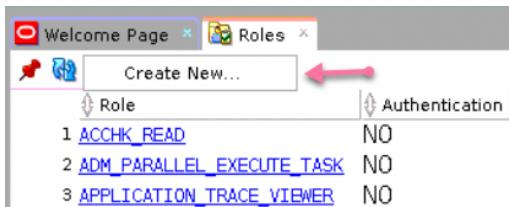
For Instructor Use Only.
This document should not be distributed.

3. Expand **Security** and then select **Roles**.



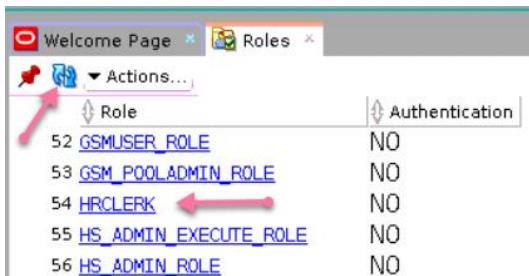
4. In the Roles tab, select Actions > Create New ...

Role	Authentication
1 ACCHK_READ	NO
2 ADM_PARALLEL_EXECUTE_TASK	NO
3 APPLICATION_TRACE_VIEWER	NO
4 AQ_ADMINISTRATOR_ROLE	NO
5 AQ_USER_ROLE	NO
6 AUDIT_ADMIN	NO
7 AUDIT_VIEWER	NO
8 AUTHENTICATEDUSER	NO
9 AVTUNE_PKG_ROLE	NO
10 BDSQL_ADMIN	NO
11 BDSQL_USER	NO
12 CAPTURE_ADMIN	NO
13 CDB_DBA	NO
14 CONNECT	NO
15 CTXAPP	NO
16 DATAPUMP_CLOUD_EXP	NO
17 DATAPUMP_CLOUD_IMP	NO
18 DATAPUMP_EXP_FULL_DATABASE	NO
19 DATAPUMP_IMP_FULL_DATABASE	NO
20 DBA	NO
21 DBFS_ROLE	NO
22 DBJAVASCRIPT	NO
...	



5. In the Create Role box,
 - a. On the **Roles** tab, enter Role Name: **HRCLERK**.
Note: Leave Password fields blank.
 - b. Click the **SQL** tab to view the SQL statement.
 - c. Click **Apply**.
 - d. In the Successful box, click **OK**.
 - e. In the Create Role box, click **Close**.
6. Verify that the **HRCLERK** role is listed in the table.

Note: You may have to click refresh



Add Object Privileges to the HRCLERK Role

- In the Connections box, expand **orclpdb1> Other USERS> HR > Tables** and then click **Employees**.

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections tree shows various users, including ANONYMOUS, APPQOSSYS, AUDSYS, BI, C##CDB_ADMIN1, CTXSYS, DBSPFWUSER, DBSNMP, DGPDB_INT, DIP, DVF, DVSY, GGSHAREDCA, GGSYS, GSMADMIN_INTERNAL, GSMCATUSER, GSMUSER, and HR. The HR schema is expanded, showing its tables: COUNTRIES, DEPARTMENTS, EMPLOYEES, JOB_HISTORY, JOBS, LOCATIONS, and REGIONS. A pink arrow points to the EMPLOYEES table. On the right, the EMPLOYEES table structure is displayed with columns: COLUMN_NAME, DATA_TYPE, and NULLABLE. The columns listed are: 1 EMPLOYEE_ID (NUMBER(6,0)), 2 FIRST_NAME (VARCHAR2(20 BYTE)), 3 LAST_NAME (VARCHAR2(25 BYTE)), 4 EMAIL (VARCHAR2(25 BYTE)), 5 PHONE_NUMBER (VARCHAR2(20 BYTE)), 6 HIRE_DATE (DATE), 7 JOB_ID (VARCHAR2(10 BYTE)), 8 SALARY (NUMBER(8,2)), 9 COMMISSION_PCT (NUMBER(2,2)), 10 MANAGER_ID (NUMBER(6,0)), and 11 DEPARTMENT_ID (NUMBER(4,0)).

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

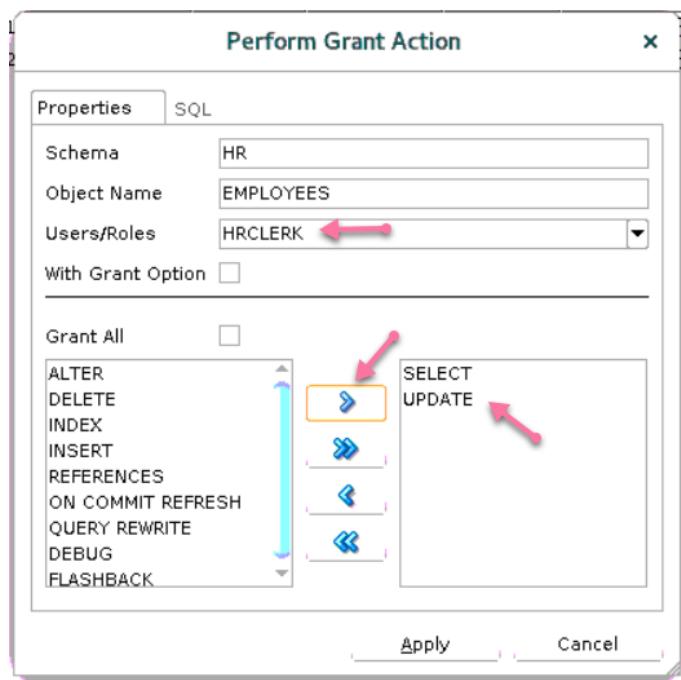
The screenshot shows the EMPLOYEES table details in Oracle SQL Developer. The Grants subtab is selected, indicated by a pink arrow. The table lists two grants: 1. SELECT privilege granted to OE with GRANTABLE set to NO, and 2. REFERENCES privilege granted to OE with GRANTABLE set to NO. The grants are issued by the HR user to the EMPLOYEES table.

PRIVILEGE	GRANTEE	GRANTABLE	GRANTOR	OBJECT_NAME
1 SELECT	OE	NO	HR	EMPLOYEES
2 REFERENCES	OE	NO	HR	EMPLOYEES

- Click **Actions > Privileges > Grant**.

- The Perform Grant Action box is displayed.
- In the Users/Roles drop-down list, select **HRCLERK**.
- In the left-hand list, click **SELECT** and move it to the right-hand list.

- d. In the left-hand list, click **UPDATE** and move it to the right-hand list.



- e. Click the SQL tab to view the SQL statement.

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

- f. Click **Apply**.
 g. In the Successful box, click **OK**.
 h. Refresh on the Grants subtab and verify **SELECT** and **UPDATE** on **HR.EMPLOYEES** have been granted to **HRCLERK**.

PRIVILEGE	GRANTEE	GRANTABLE	GRANTOR	OBJECT_NAME
1 SELECT	OE	NO	HR	EMPLOYEES
2 SELECT	HRCLERK	NO	HR	EMPLOYEES
3 UPDATE	HRCLERK	NO	HR	EMPLOYEES
4 REFERENCES	OE	NO	HR	EMPLOYEES

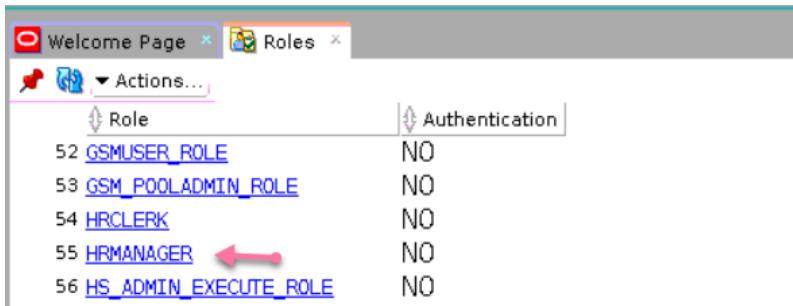
Create the HRMANAGER Role

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

10. Expand the `orclpdb1` connection in the DBA box.
11. Expand **Security** and then select **Roles**.
12. On the Roles tab, select **Actions > Create New ...**

13. In the Create Role box,
 - a. On the Roles tab, enter Role Name: **HRMANAGER**.
 - b. Click the SQL tab to view the SQL statement.
 - c. Click **Apply**.
 - d. In the Successful box, click **OK**.
 - e. In the Create Role box, click **Close**.

14. Verify that the **HRMANAGER** role is listed in the table.

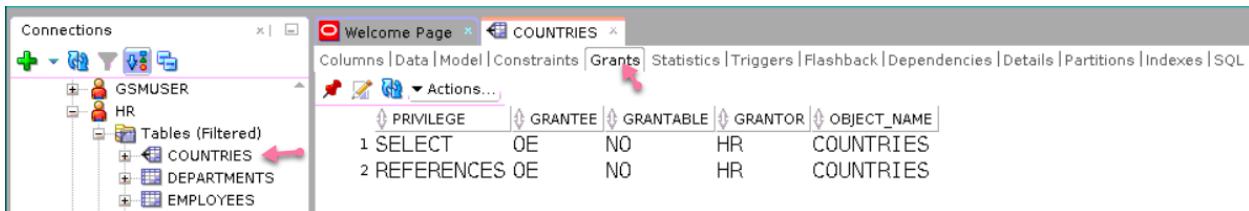


The screenshot shows the Oracle Database Roles window. The table lists roles with their IDs and names. A red arrow points to the row for role ID 55, which is **HRMANAGER**.

Role	Authentication
52 GSMUSER_ROLE	NO
53 GSM_POOLADMIN_ROLE	NO
54 HRCLERK	NO
55 HRMANAGER ←	NO
56 HS_ADMIN_EXECUTE_ROLE	NO

Add Object Privileges to HRMANAGER role

15. In the Connections box, expand **orclpdb1> Other USERS> HR> Tables** and click **COUNTRIES**.
16. Click the **Grants** subtab.

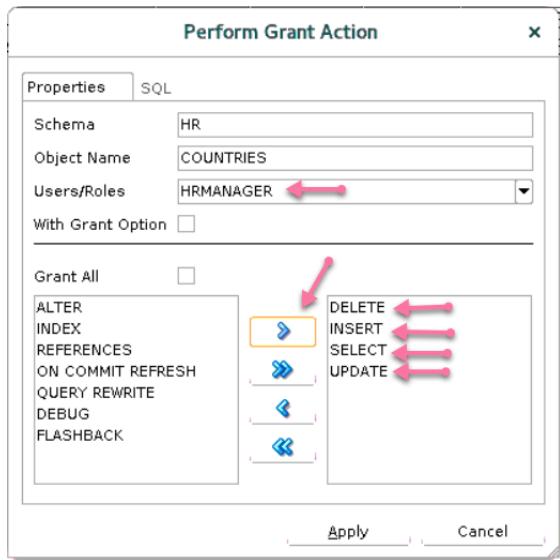


The screenshot shows the Oracle Database Tables window. The left pane shows connections and tables. A red arrow points to the **COUNTRIES** table under the HR connection. The right pane shows the grants for the COUNTRIES table. A red arrow points to the **Grants** subtab. The grants table has the following data:

PRIVILEGE	GRANTEE	GRANTABLE	GRANTOR	OBJECT_NAME
1 SELECT	OE	NO	HR	COUNTRIES
2 REFERENCES	OE	NO	HR	COUNTRIES

17. Click **Actions > Privileges > Grant**.
 - a. The Perform Grant Action box is displayed.
 - b. In the Users/Roles drop-down list, select **HRMANAGER**.
 - c. In the left-hand list, click **DELETE** and move it to the right-hand list.

- d. Repeat for **INSERT**, **SELECT**, and **UPDATE**.



- e. Click the SQL tab to view the SQL statement.

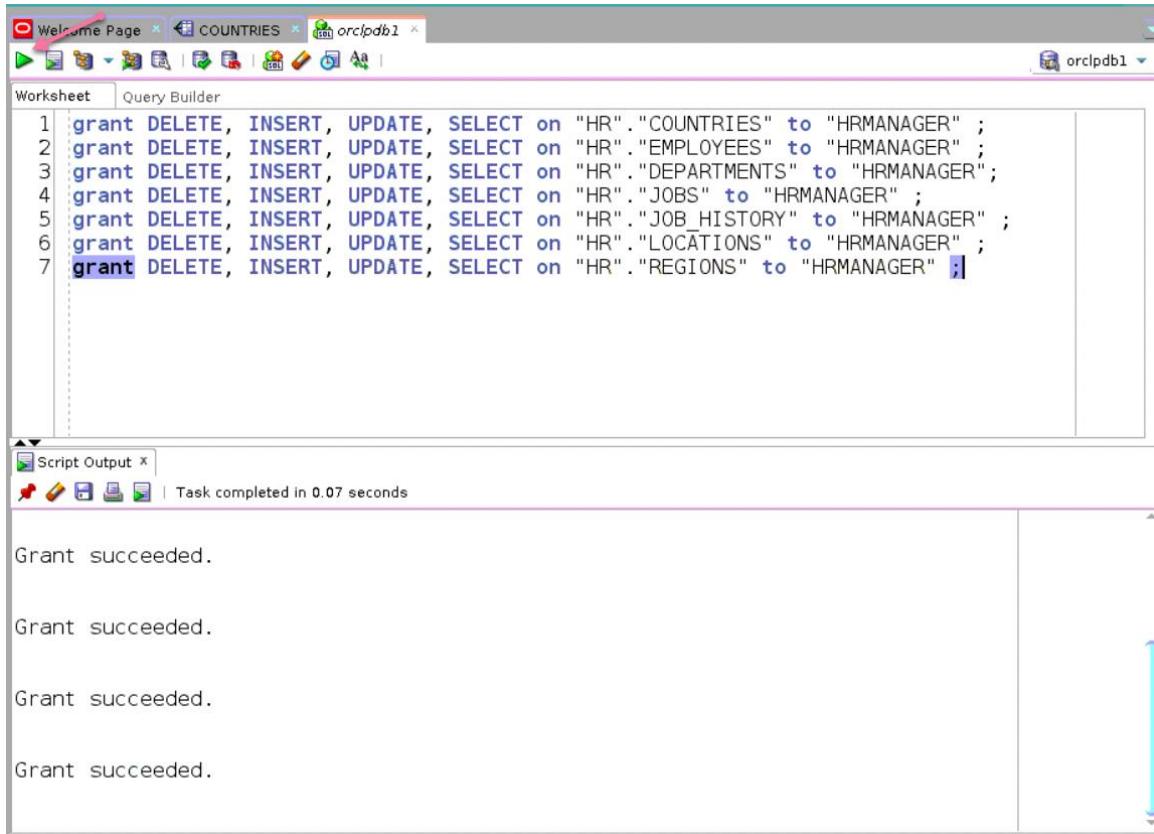
```
grant DELETE, INSERT, UPDATE, SELECT on "HR"."COUNTRIES" to
"HRMANAGER" ;
```

- f. Highlight the SQL statement and copy it (use the middle mouse button or ctrl-V).
 g. Cancel the Perform Grant Action box; click **Cancel**.
 h. Click the **PDB1-pdbadmin** tab for the Worksheet. Note that if you closed the tab, you can launch a new Worksheet using ALT-F10 or select from the SQL Developer menu **Tools > SQLWorksheet**.
 i. Paste the SQL Statement into the worksheet (use the middle mouse button or ctrl-V).
 j. Press **Ctrl-Enter** or the green right arrow to execute the statement.
 k. Change **COUNTRIES** in the statement to **EMPLOYEES** and press the green right arrow in the menu to execute.

Note: Since the table name is in quotes, upper case is required.

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

- I. Repeat for changing the table name to **DEPARTMENTS**, **JOBS**, **JOB_HISTORY**, **LOCATIONS**, **REGIONS**. Execute all the statements by clicking the run script  button. Note the script output window:



The screenshot shows the Oracle SQL Developer interface. In the top navigation bar, there are tabs for 'Welcome Page', 'COUNTRIES', and 'orclpdb1'. Below the tabs is a toolbar with various icons. The main area is titled 'Worksheet' and contains a 'Query Builder' tab. Inside the query builder, seven grants are listed:

```

1 grant DELETE, INSERT, UPDATE, SELECT on "HR"."COUNTRIES" to "HRMANAGER";
2 grant DELETE, INSERT, UPDATE, SELECT on "HR"."EMPLOYEES" to "HRMANAGER";
3 grant DELETE, INSERT, UPDATE, SELECT on "HR"."DEPARTMENTS" to "HRMANAGER";
4 grant DELETE, INSERT, UPDATE, SELECT on "HR"."JOBS" to "HRMANAGER";
5 grant DELETE, INSERT, UPDATE, SELECT on "HR"."JOB_HISTORY" to "HRMANAGER";
6 grant DELETE, INSERT, UPDATE, SELECT on "HR"."LOCATIONS" to "HRMANAGER";
7 grant DELETE, INSERT, UPDATE, SELECT on "HR"."REGIONS" to "HRMANAGER";

```

Below the worksheet is a 'Script Output' window. It displays four lines of text: 'Grant succeeded.' repeated four times. At the bottom of the output window, it says 'Task completed in 0.07 seconds'.

18. Verify the **HRMANAGER** role and the required privileges.
- In the DBA box, expand **Security** and click **Roles**.
 - Double-click the **HRMANAGER** role in the view on the right.
 - In the **HRMANAGER** tab, click the **Object Prvs** subtab.

- d. Verify the privileges granted on the HR tables are shown.

The screenshot shows the SQL*Developer interface with the 'Object Prv's tab selected in the top navigation bar. A red arrow points to the 'Object Prv's tab. Another red arrow points to the 'Object Privilege' column header in the main grid. A third red arrow points to the 'Schema' column header. A fourth red arrow points to the 'Object' column header. A fifth red arrow points to the 'Roles' node in the DBA tree on the left. A sixth red arrow points to the 'Security' node under 'Roles'. A seventh red arrow points to the 'Audit Settings' node under 'Security'.

	Object Privilege	Schema	Object
1	DELETE	HR	COUNTRIES
2	INSERT	HR	COUNTRIES
3	SELECT	HR	COUNTRIES
4	UPDATE	HR	COUNTRIES
5	DELETE	HR	DEPARTMENTS
6	INSERT	HR	DEPARTMENTS
7	SELECT	HR	DEPARTMENTS
8	UPDATE	HR	DEPARTMENTS
9	DELETE	HR	EMPLOYEES
10	INSERT	HR	EMPLOYEES
11	SELECT	HR	EMPLOYEES
12	UPDATE	HR	EMPLOYEES
13	DELETE	HR	JOB_HISTORY
14	INSERT	HR	JOB_HISTORY
15	SELECT	HR	JOB_HISTORY
16	UPDATE	HR	JOB_HISTORY
17	DELETE	HR	LOCATIONS
18	INSERT	HR	LOCATIONS
19	SELECT	HR	LOCATIONS
20	UPDATE	HR	LOCATIONS
21	DELETE	HR	REGIONS
22	INSERT	HR	REGIONS
23	SELECT	HR	REGIONS
24	UPDATE	HR	REGIONS
25	DELETE	HR	REGIONS
26	INSERT	HR	REGIONS
27	SELECT	HR	REGIONS
28	UPDATE	HR	REGIONS

19. Exit SQL*Developer by clicking **File > Exit** (**Note:** If you receive a pop-up message to save your worksheet, press cancel.)

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 20: Configuring User Resource Limits and Default Role

Overview

In these practices, you will create and manage user profiles and set a user's default role.

For Instructor Use Only.
This document should not be distributed.

Practice 20-1: Using SQL*Developer to Create a Local Profile

Overview

In this practice, you will create a local profile called `HRRPROFILE` in `ORCLPDB1` 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 ends.

In addition, the profile will be configured to automatically lock a database user account if it did not log in 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.

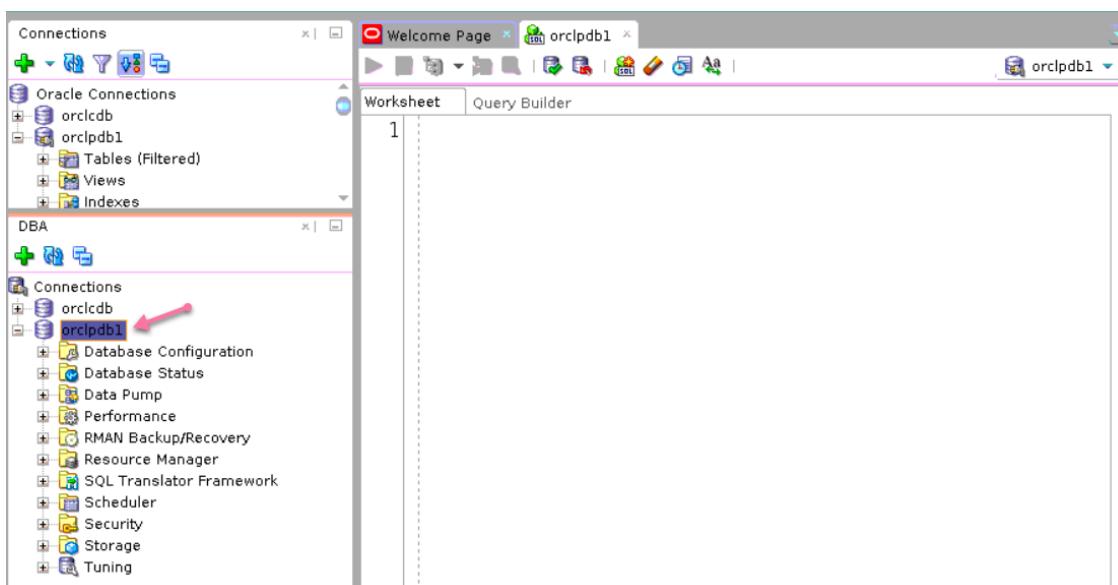
Assumptions

You are currently logged in as the `oracle` OS user.

Tasks

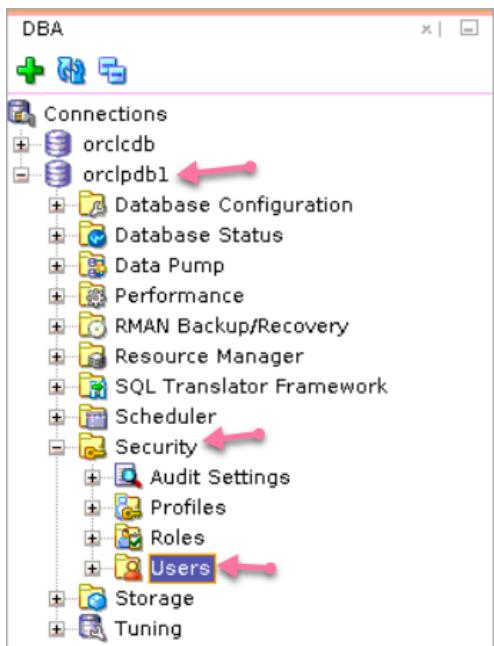
Log in to SQL*Developer (ORCLPDB1)

1. Launch SQL*Developer.
2. Click Menu View > DBA.
 - a. In the DBA box, double-click: `orclpdb1`.
 - b. A worksheet for `orclpdb1` will appear on the right.



View Privileges and Roles for PDBADMIN user

- In the DBA box, expand orclpdb1 > Security> Users.



- On the Users tab, select PDBADMIN.

User Name	Account Status	Expiration Date	Default Tablespace	Temporary Tablespace
ORCLCDB\$	LOCKED	(null)	SYSAUX	TEMP
14_GGSYS	LOCKED	(null)	SYSAUX	TEMP
15_GSMADMIN_INTERNAL	LOCKED	(null)	USERS	TEMP
16_GSMCATUSER	LOCKED	(null)	USERS	TEMP
17_GSMUSER	LOCKED	(null)	USERS	TEMP
18_HR	OPEN	07-AUG-24	USERS	TEMP
19_IX	OPEN	07-AUG-24	USERS	TEMP
20_LBACSYS	LOCKED	(null)	SYSTEM	TEMP
21_MDDATA	LOCKED	(null)	USERS	TEMP
22_MDSYS	LOCKED	(null)	SYSAUX	TEMP
23_OE	OPEN	07-AUG-24	USERS	TEMP
24_OJVMSYS	LOCKED	(null)	USERS	TEMP
25_OLAPSYS	LOCKED	(null)	SYSAUX	TEMP
26_OUTLN	LOCKED	(null)	SYSTEM	TEMP
27_PDB1_ADMIN	OPEN	26-AUG-24	USERS	TEMP
28_PDBADMIN	OPEN	07-AUG-24	USERS	TEMP
29_PM	OPEN	07-AUG-24	USERS	TEMP

5. The PDBADMIN tab is displayed. Click the **Roles** subtab.

Role	Admin Option	Default
1 DBA	NO	YES
2 PDB_DBA	YES	YES

6. In the DBA box, expand **Security** and **Roles** and select **DBA**.

Role	Authentication
1 ACCHK_READ	NO
2 ADM_PARALLEL_EXECUTE_TASK	NO
3 APPLICATION_TRACE_VIEWER	NO
4 AQ_ADMINISTRATOR_ROLE	NO
5 AQ_USER_ROLE	NO
6 AUDIT_ADMIN	NO
7 AUDIT_VIEWER	NO
8 AUTHENTICATEDUSER	NO
9 AVTUNE_PKG_ROLE	NO
10 BDSQL_ADMIN	NO
11 BDSQL_USER	NO
12 CAPTURE_ADMIN	NO
13 CDB_DBA	NO
14 CONNECT	NO
15 CTXAPP	NO
16 DATAPUMP_CLOUD_EXP	NO
17 DATAPUMP_CLOUD_IMP	NO
18 DATAPUMP_EXP_FULL_DATABASE	NO
19 DATAPUMP_IMP_FULL_DATABASE	NO
20 DBA	NO
21 DBFS_ROLE	NO
22 DBJAVASCRIPT	NO

- In the DBA tab, click the **Sys Prvs** subtab. Scroll down through the list of privileges.

System Privilege	Admin Option
1 ADMINISTER ANY SQL TUNING SET	NO
2 ADMINISTER DATABASE TRIGGER	NO
3 ADMINISTER FINE GRAINED AUDIT POLICY	NO
4 ADMINISTER REDACTION POLICY	NO
5 ADMINISTER RESOURCE MANAGER	NO
6 ADMINISTER ROW LEVEL SECURITY POLICY	NO
7 ADMINISTER SQL MANAGEMENT OBJECT	NO
8 ADMINISTER SQL TUNING SET	NO
9 ADVISOR	NO

- Hold the cursor 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**.

Filter: System Privilege

System Privilege	Admin Option
1 ADMINISTER ANY SQL TUNING SET	NO
2 ADMINISTER DATABASE TRIGGER	NO
3 CREATE PROFILE	NO

System Privilege	Admin Option
1 CREATE PROFILE	NO

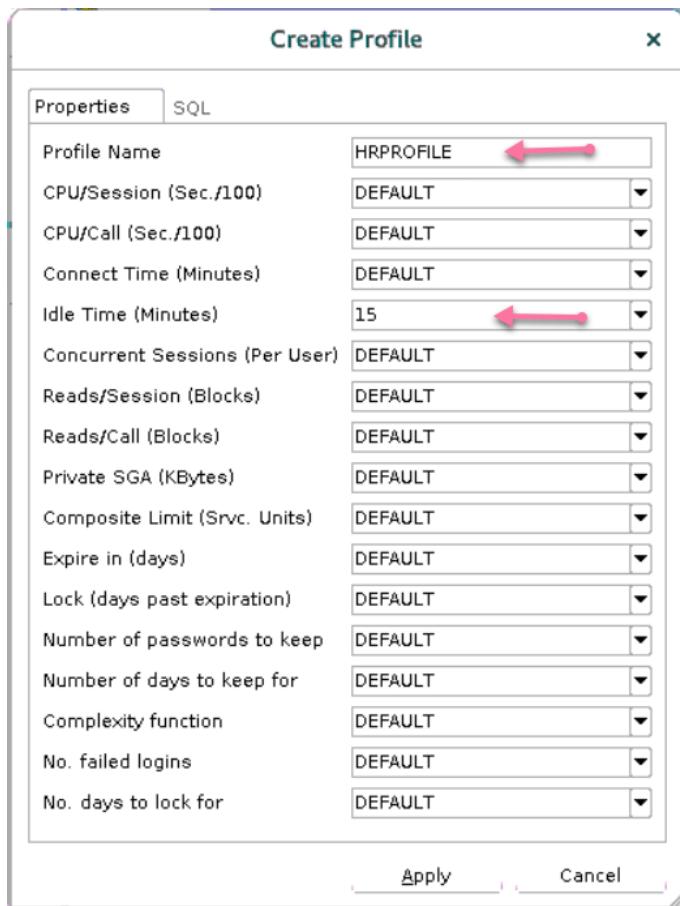
- Close the DBA Role tab.

Create a Local Profile

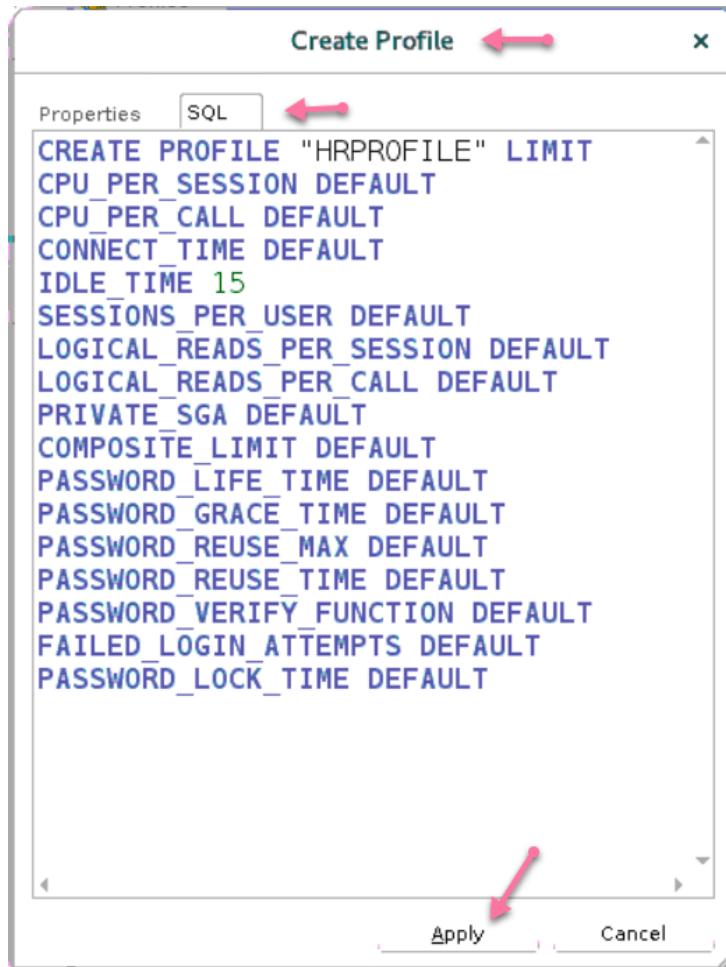
- In the DBA box, expand **Security** and click **Profiles**.
- In the Profiles tab, click **Actions > Create New ...**

12. In the Create Profile box, enter:
- Profile Name: **HRPROFILE**
 - Idle Time (minutes): **15**

Leave all other fields set to Default.



- Click the **SQL** tab to review the SQL command for this task. When done, click **APPLY**.



- In the Successful window, click **OK**.
- Verify that **HRPROFILE** is in the list of profiles.

Note: You may need to click **Refresh** .

Profile	Connect Time (Min.)	Concurrent Sessions
1 DEFAULT	UNLIMITED	UNLIMITED
2 HRPROFILE	DEFAULT	DEFAULT
3 ORA_CIS_PROFILE	DEFAULT	10
4 ORA_STIG_PROFILE	DEFAULT	DEFAULT

Set the RESOURCE_LIMIT Initialization Parameter

16. In the DBA box, expand **Database Configuration** and select **Initialization Parameters**.
17. Hold the cursor over the Parameter header and click the Filter icon. Type **resource_limit** in the Filter: Parameter field and press **Enter**.

The screenshot shows the SQL*Developer interface with two panes. The left pane displays database connections and objects under 'Connections' and 'DBA'. The right pane shows the 'Initialization Parameters' table. A red arrow points to the 'Filter: Parameter' input field at the top of the table, which contains 'resource_limit'. Another red arrow points to the 'Parameter' column header. The table lists various initialization parameters with their values. The 'resource_limit' row is highlighted.

Parameter	Value
PREPRC_140	FALSE
instance_recovery_bloom_filter_size	1048576
resource_limit	(null)
adg_account_info_tracking	LOCAL
adg_redirect_dml	FALSE
allow_global_dblinks	FALSE
allow_group_access_to_sga	FALSE
allow_legacy_reco_protocol	TRUE
allow_rowid_column_type	FALSE
allow_weak_crypto	TRUE
approx_for_aggregation	FALSE
approx_for_count_distinct	FALSE

18. The RESOURCE_LIMIT initialization parameter is listed in the table. Verify that the RESOURCE LIMIT value is set to TRUE.
19. If RESOURCE_LIMIT is not set to true, perform the following steps:
 - a. Double-click the Value field of the RESOURCE_LIMIT row.
 - b. Select TRUE and click somewhere outside the Value field.
 - c. The row number will have an asterisk; then you can press F11 or click the icon to commit the change.
 - d. In the Commit Strategy dialog box, check both **Memory** and **SPFILE** boxes and then click **Apply**.
20. Close the SQL*Developer window.

Modify the Profile So That Database User Accounts Will be Locked If Not Used in 10 Days

To lock database user accounts, modify the **HRPROFILE** profile to add the **INACTIVE_ACCOUNT_TIME** user resource profile limit. In this section, you will use SQL*Plus and learn by trial and error.

21. Open a new terminal window and connect to ORCLPDB1 as system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1
...
SQL>
```

22. 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
Help: https://docs.oracle.com/error-help/db/ora-02377/

SQL>
```

23. **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_ROLLOVER_TIME DEFAULT
PASSWORD PASSWORD_VERIFY_FUNCTION DEFAULT

18 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.

24. Investigate by displaying the full error message that you received in step 4. To do this, issue the oerr OS command for the error number ora 2377.

Note: 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:
//           * The specified limit of the PASSWORD_ROLLOVER_TIME profile
//             parameter cannot exceed either 60 days, or the current
//             value of the PASSWORD_GRACE_TIME limit of the profile,
//             or the current value of the PASSWORD_LIFE_TIME limit of
//             the profile; whichever value is lowest.
//           * When the PASSWORD_ROLLOVER_TIME profile parameter is
//             non-zero (the password rollover feature is enabled),
//             the specified limit cannot be less than 1 hour, to give
//             the application time to switch over to the new password.
//           * The PASSWORD_LIFE_TIME and PASSWORD_GRACE_TIME limits
//             cannot be less than the PASSWORD_ROLLOVER_TIME limit.
//           * For the INACTIVE_ACCOUNT_TIME profile parameter, the
//             specified
//               limit cannot be less than 15 days.
//           * For the PASSWORD_GRACE_TIME profile parameter, 0 is
allowed
//           as a permissible value.
SQL>
```

25. 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>
```

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

```
SQL> SELECT resource_type, resource_name, limit FROM dba_profiles
WHERE profile='HRPROFILE' ORDER BY 1,2;

RESOURCE RESOURCE_NAME                      LIMIT
-----
KERNEL  COMPOSITE_LIMIT                     DEFAULT
KERNEL  CONNECT_TIME                        DEFAULT
KERNEL  CPU_PER_CALL                       DEFAULT
KERNEL  CPU_PER_SESSION                     DEFAULT
KERNEL  IDLE_TIME                          15
KERNEL  LOGICAL_READS_PER_CALL             DEFAULT
KERNEL  LOGICAL_READS_PER_SESSION          DEFAULT
KERNEL  PRIVATE_SGA                         DEFAULT
KERNEL  SESSIONS_PER_USER                  DEFAULT
PASSWORD FAILED_LOGIN_ATTEMPTS            DEFAULT
PASSWORD INACTIVE_ACCOUNT_TIME            15
PASSWORD PASSWORD_GRACE_TIME              DEFAULT
PASSWORD PASSWORD_LIFE_TIME               DEFAULT
PASSWORD PASSWORD_LOCK_TIME               DEFAULT
PASSWORD PASSWORD_REUSE_MAX              DEFAULT
PASSWORD PASSWORD_REUSE_TIME              DEFAULT
PASSWORD PASSWORD_ROLLOVER_TIME          DEFAULT
PASSWORD PASSWORD_VERIFY_FUNCTION        DEFAULT

18 rows selected.

SQL>
```

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

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$ EXIT
```

28. **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 20-2: Configuring a Default Role for a User

Overview

In this practice, you will create and configure `jgoodman` (user account for Jenny Goodman in ORCLPDB1).

Assumptions

You will create a user account called `jgoodman`, grant CONNECT and RESOURCE roles to it, set the CONNECT role as the user's default role, and then connect as the `jgoodman` user and validate that only the default role is enabled in the session and then enable both roles for the running session.

Tasks

Configure a Default Role for JGOODMAN

1. Open a terminal window and start SQL*Plus and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1  
...  
SQL>
```

2. Create the user `jgoodman` and assign CONNECT and RESOURCE roles to it.

Note: Drop the user `jgoodman` first just in case the user exists.

```
SQL> DROP USER jgoodman CASCADE;  
  
DROP USER jgoodman CASCADE  
*  
ERROR at line 1:  
ORA-01918: user 'JGOODMAN' does not exist  
Help: https://docs.oracle.com/error-help/db/ora-01918/  
  
SQL> CREATE USER jgoodman IDENTIFIED BY WELCOME123##;  
  
User created.  
  
SQL> GRANT connect, resource TO jgoodman;  
  
Grant succeeded.
```

3. 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, `CONNECT` and `RESOURCE`, and both are default roles (the `DEF` column = `YES`).

```
SQL> COL granted_role FORMAT A20
SQL> SELECT granted_role, default_role FROM dba_role_privs WHERE
grantee='JGOODMAN';

GRANTED_ROLE          DEF
-----
CONNECT              YES
RESOURCE             YES
```

4. Set the default role for `JGOODMAN` to be `CONNECT` only by using the `ALTER USER` command and the `DEFAULT ROLE` clause.

```
SQL> ALTER USER jgoodman DEFAULT ROLE connect;
User altered.

SQL>
```

5. 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 `CONNECT` and the `RESOURCE` 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
-----
CONNECT              YES
RESOURCE             NO

SQL>
```

6. Disconnect from `ORCLPDB1`.

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

Enable a Non-Default Role

7. Connect to ORCLPDB1 as jgoodman user.

```
SQL> CONNECT jgoodman/WElcome123##@orclpdb1
Connected.
SQL>
```

8. View the roles for the current session.

Note: The default role, CONNECT, is in effect.

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

9. Suppose jgoodman needs to operate as a developer. Change the enabled roles to CONNECT and RESOURCE. **Caution:** When you use the SET ROLE command, any roles not included in the command will be disabled.

```
SQL> SET ROLE connect, resource;
Role set.
SQL>
```

10. View the roles for the current session again.

```
SQL> SELECT * FROM session_roles;
ROLE
-----
CONNECT
RESOURCE
SODA_APP
SQL>
```

Note: SODA_APP is a role included in the RESOURCE role.

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

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 21: Implementing Oracle Database Auditing

Overview

In these practices, you will verify Unified Auditing is enabled, create audit users, and create an audit policy.

For Instructor Use Only.
This document should not be distributed.

Practice 21-1: Enabling Unified Auditing

Overview

In this practice, you will enable Unified Auditing.

Assumptions

When you install a 23c Oracle Database, by default Unified Auditing is enabled; however, if you upgrade your database from previous releases and Unified Auditing is not enabled, then you must manually enable it.

To demonstrate how to enable the Unified Audits, we will remove Unified Audits from Oracle executables and remove all existing Unified Auditing artifacts by executing the following script:

```
/home/oracle/labs/DBMod_UsersSec/disable_unified_aud.sh
```

After the script concludes, only the `orclcldb` database will be restarted. This script may display errors that can be safely ignored.

Tasks

1. Open a terminal and execute

```
/home/oracle/labs/DBMod_UsersSec/disable_unified_aud.sh; then after the  
script concludes, shut down all Oracle processes of all instances running from your Oracle  
Home.
```

- a. Remove Unified Auditing from your Oracle executables.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_UsersSec/disable_unified_aud.sh  
The Oracle base remains unchanged with value /u01/app/oracle  
  
LSNRCTL for Linux: Version 23.0.0.0.0 - Limited Availability  
on 06-MAR-2024 01:50:10  
  
Copyright (c) 1991, 2023, Oracle. All rights reserved.  
  
Connecting to  
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0) (PORT=1521))  
)  
The command completed successfully  
  
SQL*Plus: Release 23.0.0.0.0 - Limited Availability on Wed Mar  
6 01:50:10 2024  
Version 23.4.0.23.11  
  
Copyright (c) 1982, 2023, Oracle. All rights reserved.
```

```
Connected to:  
...  
SQL> Disconnected from Oracle Database 23c Enterprise Edition  
Release 23.0.0.0.0 - Limited Availability  
Version 23.4.0.23.11  
/usr/bin/ar d  
/u01/app/oracle/product/23.4.0/dbhome_1/rdbms/lib/libknlopt.a  
kzaiang.o  
/usr/bin/ar cr  
/u01/app/oracle/product/23.4.0/dbhome_1/rdbms/lib/libknlopt.a  
/u01/app/oracle/product/23.4.0/dbhome_1/rdbms/lib/kzanang.o  
chmod 755 /u01/app/oracle/product/23.4.0/dbhome_1/bin  
cd /u01/app/oracle/product/23.4.0/dbhome_1/rdbms/lib/;  
...  
SQL> ORACLE instance started.  
  
Total System Global Area 2010661640 bytes  
Fixed Size 5317384 bytes  
Variable Size 1224736768 bytes  
Database Buffers 771751936 bytes  
Redo Buffers 8855552 bytes  
Database mounted.  
Database opened.  
SQL> Disconnected from Oracle Database 23c Enterprise Edition  
Release 23.0.0.0.0 - Limited Availability  
Version 23.4.0.23.11  
[oracle@edvmr1p0 DBMod_UsersSec]$
```

b. Shut down LISTENER.

```
[oracle@edvmr1p0 DBMod_UsersSec]$ lsnrctl stop listener  
...  
Connecting to  
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=edvmr1p0.us.oracle.com) (PORT=1521)))  
The command completed successfully  
[oracle@edvmr1p0 DBMod_UsersSec]$
```

- c. Shut down the `orclcdb` instance and exit SQL*Plus.

```
[oracle@edvmr1p0 DBMod_UsersSec]$ sqlplus / as sysdba
...
SQL> SHUTDOWN IMMEDIATE
...
SQL> EXIT
...
[oracle@edvmr1p0 DBMod_UsersSec]$
```

2. Enable the Unified Auditing feature.

```
[oracle@edvmr1p0 DBMod_UsersSec]$ cd $ORACLE_HOME/rdbms/lib
[oracle@edvmr1p0 lib]$ make -f ins_rdbms.mk uniaud_on ioracle
...
[oracle@edvmr1p0 lib]$
```

3. Restart the processes.

- a. Restart the listener.

```
[oracle@edvmr1p0 lib]$ lsnrctl start listener
...
[oracle@edvmr1p0 lib]$
```

- b. Restart the `orclcdb` database instance.

```
[oracle@edvmr1p0 lib]$ sqlplus / as sysdba
...
Connected to an idle instance.
SQL> STARTUP
...
SQL>
```

4. Verify that Unified Auditing is enabled.

```
SQL> COL parameter FORMAT A20
SQL> COL value FORMAT A20
SQL> SELECT * FROM v$option WHERE PARAMETER = 'Unified Auditing';

PARAMETER          VALUE          CON_ID
-----  -----
Unified Auditing    TRUE           0
SQL>
```

5. Open all PDBs.

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

6. Set all databases to open on startup.

```
SQL> ALTER PLUGGABLE DATABASE all SAVE STATE;  
  
Pluggable database altered.  
  
SQL>
```

7. Exit from SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 lib]$
```

8. Exit all terminals.

Practice 21-2: 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 as it 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.

Tasks

1. Open a terminal window and connect to the ORCLCDB instance as a user with the SYSDBA privilege.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba  
...  
SQL>
```

2. Create a common database user to be the administrator of the audit settings and policies. Name this user as c##audmgr and grant it CONNECT and AUDIT_ADMIN roles.

```
SQL> CREATE USER c##audmgr IDENTIFIED BY WELCOME123##  
  container=all;  
  
User created.  
  
SQL> GRANT connect, audit_admin TO c##audmgr container = all;  
  
Grant succeeded.  
  
SQL>
```

3. Create a database user to be used by any person that needs to view the audit data. Name this user as c##audvwr and grant it CONNECT and AUDIT_VIEWER roles. 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 WELCOME123##  
  container=all;  
  
User created.  
  
SQL> GRANT connect, audit_viewer TO c##audvwr container=all;
```

```
Grant succeeded.
```

```
SQL>
```

4. Exit SQL*Plus.

```
SQL> EXIT
```

```
...
```

```
[oracle@edvmr1p0 ~] $
```

5. Close all terminals.

Practice 21-3: Creating an Audit Policy

Overview

In this practice, as the C##AUDMGR user you will create an audit policy to monitor activity in the PDBADMIN.EMP table in the ORCLPDB1 database and apply it to multiple users.

Tasks

1. **(Catchup Step)** If you *skipped* the previous practice **21-2** in this lesson, then run the following script: /home/oracle/labs/DBMod_UsersSec/setup_users_lab22.sh. Otherwise, proceed to **step 2**.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_UsersSec/setup_users_lab22.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. Open a terminal window and create the EMP table in the hr schema and insert sample data into it.

```
[oracle@edvmr1p0 ~]$ sqlplus hr/WELCOME123##@orclpdb1  
...  
SQL> CREATE TABLE emp  
      (id number, name varchar2(20), salary number, job varchar2(5),  
       depno number);  
  
Table created.  
  
SQL> ALTER TABLE emp ADD CONSTRAINT emp_id_pk PRIMARY KEY (id);  
  
Table altered.  
  
SQL> INSERT INTO emp VALUES (&id,'&name',&sal,'&job',&depno);  
Enter value for id: 101  
Enter value for name: King  
Enter value for sal: 12000  
Enter value for job: MAN  
Enter value for depno: 10  
old   1: INSERT INTO emp VALUES (&id,'&name',&sal,'&job',&depno)  
new   1: INSERT INTO emp VALUES (101,'King',12000,'MAN',10)
```

```
1 row created.

SQL> /
Enter value for id: 102
Enter value for name: Jhon
Enter value for sal: 15000
Enter value for job: PRES
Enter value for depno: 10
old    1: INSERT INTO emp VALUES (&id,'&name',&sal,'&job',&depno)
new    1: INSERT INTO emp VALUES (102,'Jhon',15000,'PRES',10)

1 row created.

SQL> /
Enter value for id: 103
Enter value for name: Neena
Enter value for sal: 18000
Enter value for job: HR
Enter value for depno: 20
old    1: INSERT INTO emp VALUES (&id,'&name',&sal,'&job',&depno)
new    1: INSERT INTO emp VALUES (103,'Neena',18000,'HR',20)

1 row created.

SQL> COMMIT;

Commit complete.

SQL>
```

3. Connect to the ORCLPDB1 database as the c##audmgr user and create a policy named EMP_AUDIT_UPD that audits update activity in the HR.EMP table.
 - a. Connect to the ORCLPDB1 PDB as the c##audmgr user by using SQL*Plus.

```
SQL> CONN c##audmgr/WElcome123##@orclpdb1
...
SQL>
```

- b. Create an audit policy called `EMP_AUDIT_UPD` to track UPDATE commands issued against the `HR.EMP` table.

```
SQL> CREATE AUDIT POLICY emp_audit_upd ACTIONS update ON
hr.emp;
```

Audit policy created.

```
SQL>
```

- c. Enable the audit policy for all users.

```
SQL> AUDIT POLICY emp_audit_upd;
```

Audit succeeded.

```
SQL>
```

4. Verify the creation of the `EMP_AUDIT_UPD` policy.

```
SQL> COL audit_option FORMAT A20
SQL> COL policy_name FORMAT A18
SQL> COL object_name FORMAT A18
SQL> SELECT policy_name, audit_option, object_name FROM
audit_unified_policies WHERE policy_name = 'EMP_AUDIT_UPD';
```

POLICY_NAME	AUDIT_OPTION	OBJECT_NAME
EMP_AUDIT_UPD	UPDATE	EMP

```
SQL>
```

5. Test the audit policy by connecting as a user that has privileges to update rows in the `HR.EMP` table.

- a. Connect as the `system` user, update `hr.emp`, set `SALARY` to \$50000 for `id 101`, and `COMMIT` the update.

```
SQL> CONNECT system/WElcome123##@orclpdb1
...
SQL> UPDATE HR.emp SET salary = 50000 WHERE id=101;

1 row updated.

SQL> COMMIT;
```

```
Commit complete.
```

```
SQL>
```

- b. Connect to ORCLPDB1 as the c##audmgr user flushes the audits.

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.

```
SQL> CONNECT c##audmgr/WElcome123##@orclpdb1
...
SQL> EXEC sys.dbms_audit_mgmt.flush_unified_audit_trail

PL/SQL procedure successfully completed.

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 ('SYSTEM')
   AND ACTION_NAME not in ('LOGON', 'LOGOFF')
 ORDER BY 4;

UNIFIED_AUDIT_POLICIES DBUSERNA ACTION_N DATE
-----
ORA_SECURECONFIG      JGOODMAN SET ROLE 11-APR-21 05:49
                      JGOODMAN SET ROLE 11-APR-21 05:50
ORA_SECURECONFIG      JGOODMAN SET ROLE 11-APR-21 05:50
EMP_AUDIT_UPD        JGOODMAN UPDATE    11-APR-21 06:29
```

- c. Connect to ORCLPDB1 as c##audvwr and view the audit records by querying the unified_audit_trail view.

Note: Your result may differ from the following output.

```
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 ('SYSTEM')
AND ACTION_NAME in ('GRANT', 'UPDATE')
ORDER BY 4;

2      3      4      5      6
UNIFIED_AUDIT_POLICIES DBUSERNA ACTION_N DATE
-----
ORA_SECURECONFIG        SYSTEM    GRANT    06-MAR-24 01:00
EMP_AUDIT_UPD          SYSTEM    UPDATE   06-MAR-24 03:41
ORA_SECURECONFIG        SYSTEM    GRANT    09-FEB-24 10:11
ORA_SECURECONFIG        SYSTEM    GRANT    09-FEB-24 10:11
...
44 rows selected.

SQL>

```

6. Exit from SQL*Plus.

```

SQL> EXIT
...
[oracle@edvmr1p0 ~]$

```

7. Run the /home/oracle/labs/DBMod_UsersSec/cleanup_unified_aud.sh script to reset your environment.

```

[oracle@edvmr1p0
~]$ /home/oracle/labs/DBMod_UsersSec/cleanup_unified_aud.sh
...
SQL>
Table dropped.

SQL> Disconnected from Oracle Database 23c Enterprise Edition
Release 23.0.0.0.0 - Limited Availability
Version 23.4.0.23.11
[oracle@edvmr1p0 ~]$

```

8. Close all terminal windows.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 22: Introduction to Loading and Transporting Data

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 23: Loading Data

Overview

In these practices, you will use SQL*Loader to load data.

For Instructor Use Only.
This document should not be distributed.

Practice 23-1: Loading Data into a PDB from an External File

Overview

In this practice, you will 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 execute the `$HOME/labs/DBMod_LoadTrans/DP_setup.sh` shell script.

Note: This script takes approximately 2 minutes to run.

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_LoadTrans/DP_setup.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. View the `products.dat` file to learn about its structure.

```
[oracle@edvmr1p0 ~]$ 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  
[oracle@edvmr1p0 ~]$
```

3. Start SQL*Plus and connect to ORCLPDB1 as the sh user.

```
[oracle@edvmr1p0 ~]$ sqlplus sh/WElcome123##@orclpdb1  
...  
SQL>
```

4. 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>
```

5. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

6. Change to the /home/oracle/labs/DBMod_LoadTrans directory and verify.

```
[oracle@edvmr1p0 ~]$ cd /home/oracle/labs/DBMod_LoadTrans  
[oracle@edvmr1p0 DBMod_LoadTrans]$ pwd  
/home/oracle/labs/DBMod_LoadTrans  
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

7. 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.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$  
sqlldr sh/WElcome123##@orclpdb1 table=products  
SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Wed Mar  
6 21:06:46 2024  
Version 23.4.0.23.11  
  
Copyright (c) 1982, 2023, Oracle and/or its affiliates. All  
rights reserved.  
  
Express Mode Load, Table: PRODUCTS  
Path used: External Table, DEGREE_OF_PARALLELISM=AUTO
```

```
Table PRODUCTS:
```

```
8 Rows successfully loaded.
```

```
Check the log files:
```

```
products.log
```

```
products_%p.log_xt
```

```
for more information about the load.
```

```
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

8. Start SQL*Plus and connect to ORCLPDB1 as the sh user.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$  

sqlplus sh/WElcome123##@orclpdb1  

...
SQL>
```

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

```
SQL> SELECT * FROM products;

PRODUCT_ID COU LABEL          DETAILED_LABEL
----- -----
 1001 ENG Shutter1    Outdoor shutter1
 1002 FRE Porte1      Porte exterieure1
 1003 SPA Puerta1     Puerta exterior1
 1004 GER Tur1        Auberliche Tur1
 1005 FRE Volet1      Volet exterieur1
 1006 SPA Obturador1 Obturador exterior1
 1007 GER Fenster1    Fensterladen1
 4001 ENG Door        Outdoor
 4002 FRE Porte       Porte exterieure
 4003 SPA Puerta      Puerta exterior
 4004 GER Tur         Auberliche Tur
 5001 ENG Shutter     Outdoor shutter
 5002 FRE Volet       Volet exterieur
 5003 SPA Obturador   Obturador exterior
 5004 GER Fenster     Fensterladen

15 rows selected.

SQL>
```

10. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 DBMod LoadTrans]$
```

11. View the products.log file.

```
[oracle@edvmr1p0 DBMod LoadTrans]$ cat products.log

SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Wed Mar
6 21:06:46 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, 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 Mar 06 21:06:46 2024
Run ended on Wed Mar 06 21:06:48 2024

Elapsed time was:      00:00:02.05
CPU time was:          00:00:00.03
[oracle@edvmr1p0 ~]$

```

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.

12. 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.

```

[oracle@edvmr1p0 DBMod_LoadTrans]$ ls -lt products_*.log_xt
-rw-r--r-- 1 oracle oinstall 854 Mar 6 21:06
products_692256.log_xt
[oracle@edvmr1p0 DBMod_LoadTrans]$ cat products_692256.log_xt

LOG file opened at 03/06/24 21:06:48

Total Number of Files=1

Data File: products.dat

Log File: products_692256.log_xt


LOG file opened at 03/06/24 21:06:48

Bad File: products_692256.bad

```

```

Field Definitions for table SYS_SQLLDR_X_EXT_PRODUCTS
Record format DELIMITED BY NEWLINE
Data in file has same endianness as the platform
Reject rows with all null fields

Fields in Data Source:

PRODUCT_ID           CHAR (255)
Terminated by ","
Trim whitespace from left and right
COUNTRY              CHAR (255)
Terminated by ","
Trim whitespace from left and right
LABEL                CHAR (255)
Terminated by ","
Trim whitespace from left and right
DETAILED_LABEL       CHAR (255)
Terminated by ","
Trim whitespace from left and right
[oracle@edvmr1p0 DBMod_LoadTrans]$

```

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.

13. Start SQL*Plus and connect to ORCLPDB1 as the sh user.

Note: If you started a new terminal window, please make sure that your current directory is /home/oracle/labs/DBMod_LoadTrans.

```

[oracle@edvmr1p0 DBMod_LoadTrans]$
sqlplus sh/WElcome123##@orclpdb1
...
SQL>

```

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

```

SQL> SELECT count(*) FROM inventories;

COUNT(*)
-----
476

SQL>

```

15. Exit from SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

16. Start SQL*Loader, connect to ORCLPDB1 as the sh user, and view the content of \$HOME/labs/DBMod_LoadTrans/DP_inventories_append.ctl, which will be used to load the SH.INVENTORIES table from the \$HOME/labs/DBMod_LoadTrans/DP_inventories.dat data file in conventional mode with append operation. The result shows that 83 rows were successfully loaded in the SH.INVENTORIES table.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$ cat DP_inventories_append.ctl
-- ***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
APPEND
FIELDS TERMINATED BY ','
(warehouse_id,
product_id,
quantity_on_hand)
[oracle@edvmr1p0 DBMod_LoadTrans]$

sqlldr userid=sh/WElcome123##@orclpdb1
control=DP_inventories_append.ctl log=inventories.log
data=DP_inventories.dat

SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Wed Mar
6 21:32:23 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, Oracle and/or its affiliates. All
rights reserved.

Path used:      Conventional
Commit point reached - logical record count 83

Table SH.INVENTORIES:
  83 Rows successfully loaded.

Check the log file:
  inventories.log
for more information about the load.
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

17. Start SQL*Plus and connect to ORCLPDB1 as the sh user.

Hint: Use the Linux command-line buffer recall using the up arrow.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$  
sqlplus sh/WElcome123##@orclpdb1  
...  
SQL>
```

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

```
SQL> SELECT count(*) FROM inventories;  
  
 COUNT (*)  
-----  
      559  
SQL>
```

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.

19. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

20. View the inventories.log file.

Note: The insert option in effect for the SH.INVENTORIES table is APPEND.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$ cat inventories.log  
  
SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Wed Mar  
6 21:32:23 2024  
Version 23.4.0.23.11  
  
Copyright (c) 1982, 2023, 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 Wed Mar 06 21:32:23 2024
Run ended on Wed Mar 06 21:32:24 2024

Elapsed time was:        00:00:01.60
CPU time was:            00:00:00.03
[oracle@edvmr1p0 DBMod LoadTrans]$

```

21. Now view the control file named DP_inventories_trunc.ctl in the cat command.

Note: The TRUNCATE command prior to loading data will remove all data in the table without dropping the table before loading.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$ cat DP_inventories_trunc.ctl
-- ***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)
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

22. Start SQL*Loader, connect to ORCLPDB1 as the sh user, and re-execute the load operation with the ROWS parameter set to 10.

Note: The commit points are after 10 rows and 83 rows are loaded.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$
sqlldr userid=sh/WElcome123##@orclpdb1
control=DP_inventories_trunc.ctl log=inventories.log
data=DP_inventories.dat ROWS=10

SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Wed Mar
6 21:52:48 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, Oracle and/or its affiliates. All
rights reserved.

Path used: Conventional
Commit point reached - logical record count 10
Commit point reached - logical record count 20
Commit point reached - logical record count 30
Commit point reached - logical record count 40
Commit point reached - logical record count 50
Commit point reached - logical record count 60
Commit point reached - logical record count 70
Commit point reached - logical record count 80
Commit point reached - logical record count 83

Table SH.INVENTORIES:
 83 Rows successfully loaded.
```

```
Check the log file:  
    inventories.log  
for more information about the load.  
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

23. Start SQL*Plus and connect to ORCLPDB1 as the SH user.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$  
sqlplus sh/WELCOME123##@orclpdb1  
...  
SQL>
```

24. Verify the number of rows in the INVENTORIES table. The table now has 83 rows. The TRUNCATE option cleared out the original rows in the table and inserted 83 new rows.

```
SQL> SELECT count(*) FROM inventories;  
  
COUNT (*)  
-----  
83  
SQL>
```

Re-enable the Check Constraint:

Suppose a DBA discovers that the CHECK constraint was disabled on the WAREHOUSE_ID column in the SH.INVENTORIES table at the time of the load. This disabled constraint allowed only values within a certain range. Use the \$HOME/labs/DBMod_LoadTrans/DP_check.sql SQL script to empty the table and re-enable the check constraint, then reload the table.

25. 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  
...  
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

26. Start SQL*Loader, connect to ORCLPDB1 as the SH user, and reload the table using the truncate operation.

Note: The results indicate that 20 rows were successfully loaded into the SH.INVENTORIES table.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$  
sqlldr userid=sh/WElcome123##@orclpdb1  
control=DP_inventories_trunc.ctl log=inventories.log  
data=DP_inventories.dat ROWS=10  
  
SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Wed Mar  
6 21:52:48 2024  
Version 23.4.0.23.11  
  
Copyright (c) 1982, 2023, Oracle and/or its affiliates. All  
rights reserved.  
  
Path used:          Conventional  
Commit point reached - logical record count 10  
Commit point reached - logical record count 20  
Commit point reached - logical record count 30  
Commit point reached - logical record count 40  
Commit point reached - logical record count 50  
Commit point reached - logical record count 60  
Commit point reached - logical record count 70  
Commit point reached - logical record count 80  
  
Table SH.INVENTORIES:  
  20 Rows successfully loaded.  
  
Check the log file:  
  inventories.log  
for more information about the load.  
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

27. 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.

```
[oracle@edvmr1p0 DBMod_LoadTrans]$ cat inventories.log
```

```
SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Wed Mar
6 22:02:01 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, Oracle and/or its affiliates. All
rights reserved.

Control File: DP_inventories.ctl
Data File: DP_inventories.dat
Bad File: DP_inventories.bad
Discard File: none specified

(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array: 10 rows, maximum of 1048576 bytes
Continuation: none specified
Path used: Conventional

Table SH.INVENTORIES, loaded from every logical record.
Insert option in effect for this table: TRUNCATE

      Column Name          Position   Len  Term Encl
Datatype
-----
-----
WAREHOUSE_ID           FIRST      * ,
CHARACTER
PRODUCT_ID             NEXT      * ,
CHARACTER
QUANTITY_ON_HAND       NEXT      * ,
CHARACTER

Record 21: Rejected - Error on table SH.INVENTORIES.
ORA-02290: check constraint (SH.CK_WAREHOUSE_ID) violated
Help: https://docs.oracle.com/error-help/db/ora-02290/

...
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 Wed Mar 06 22:02:01 2024
Run ended on Wed Mar 06 22:02:02 2024

Elapsed time was:        00:00:00.38
CPU time was:            00:00:00.04
[oracle@edvmr1p0 DBMod_LoadTrans]$
```

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

Answer: No. After 20 rows were successfully loaded, 51 rows did not load due to a constraint violation error. The load stopped at that 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.

28. Open a terminal window and execute the

\$HOME/labs/DBMod_LoadTrans/DP_setup.sh shell script to reset the environment for the following exercise.

Note: This script takes approximately 2 minutes to run.

```

[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_LoadTrans/DP_setup.sh
...
[oracle@edvmr1p0 ~]$
```

29. Start SQL*Loader, connect to ORCLPDB1 as the sh user, and load the SH.INVENTORIES table in direct mode using truncate operation.

Note: The results indicate that the load completed and the record count is 83.

```

[oracle@edvmr1p0 DBMod_LoadTrans]$
sqlldr userid=sh/WELCOME123##@orclpdb1
control=DP_inventories_trunc.ctl log=inventories.log
data=DP_inventories.dat ROWS=10 DIRECT=TRUE

SQL*Loader: Release 23.0.0.0.0 - Limited Availability on Thu Mar
7 22:52:08 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, Oracle and/or its affiliates. All
rights reserved.

Path used:      Direct
Save data point reached - logical record count 10.
```

```
Save data point reached - logical record count 20.
Save data point reached - logical record count 30.
Save data point reached - logical record count 40.
Save data point reached - logical record count 50.
Save data point reached - logical record count 60.
Save data point reached - logical record count 70.
Save data point reached - logical record count 80.

Load completed - logical record count 83.

Table SH.INVENTORIES:
 83 Rows successfully loaded.

Check the log file:
  inventories.log
for more information about the load.

Table SH.INVENTORIES:
 83 Rows successfully loaded.

Check the log file:
  inventories.log
for more information about the load.
[oracle@edvmr1p0 DBMod LoadTrans]$
```

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.

Question: Does SQL*Loader in direct mode enforce all constraints?

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

30. Close the terminal window.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 24: Transporting Data

Overview

In these practices, you will move data from one ORCLPDB to another PDB using multiple methods.

For Instructor Use Only.
This document should not be distributed.

Practice 24-1: Moving Data from One PDB to Another PDB

Overview

In this practice, imagine that you have 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 execute the

`$HOME/labs/DBMod_LoadTrans/DP_setup.sh` shell script to create tables in ORCLPDB1 and ORCLPDB2.

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_LoadTrans/DP_setup.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. Launch Data Pump export under a connection as oe in ORCLPDB1 to export all objects belonging to oe. Use the DUMPFILE parameter to specify the location and name of the dump file resulting from the export operation.

Note: You will get an error during this operation stating that the file name cannot contain a path specification.

```
[oracle@edvmr1p0 ~]$ expdp oe/WELCOME123##@orclpdb1 schemas=oe  
dumpfile=/u01/app/oracle/admin/orclcdb/dpdump/expoe.dmp
```

```
Export: Release 23.0.0.0.0 - Limited Availability on Wed Mar 6  
23:52:40 2024  
Version 23.4.0.23.11
```

```
Copyright (c) 1982, 2023, Oracle and/or its affiliates. All  
rights reserved.
```

```
Connected to: Oracle Database 23c Enterprise Edition Release  
23.0.0.0.0 - Limited Availability  
ORA-39001: invalid argument value  
ORA-39000: bad dump file specification  
ORA-39088: File name cannot contain a path specification.
```

```
[oracle@edvmr1p0 ~]$
```

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.

3. Start SQL*Plus and connect to ORCLPDB1 as the sys user with the SYSDBA privilege.

```
[oracle@edvmr1p0 ~]$ sqlplus sys/WELCOME123##@orclpdb1 as sysdba  
...  
SQL>
```

- Create an 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>

- 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>

- Exit SQL*Plus.

```
SQL> EXIT
```

...
[oracle@edvmr1p0 ~]\$

- Retry the Data Pump export.

```
[oracle@edvmr1p0 ~]$ expdp oe/WELCOME123##@orclpdb1 schemas=oe
directory=dp_for_oe dumpfile=expoe.dmp
...
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/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/LOGREP
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
```

```

Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
  . . exported "OE"."ORDERS"                                12.92 KB
  105 rows
  . . exported "OE"."ORDER_ITEMS"                            21.13 KB
  665 rows
Master table "OE"."SYS_EXPORT_SCHEMA_01" successfully
loaded/unloaded
*****
*****
Dump file set for OE.SYS_EXPORT_SCHEMA_01 is:
/u01/app/oracle/admin/orclcdp/dpdump/expoe.dmp
Job "OE"."SYS_EXPORT_SCHEMA_01" successfully completed at Thu Mar
7 00:00:42 2024 elapsed 0 00:01:21

[oracle@edvmr1p0 ~]$

```

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.

8. Use Data Pump Import to generate a SQL script named `oe_SQL.sql` from the dump file.

```

[oracle@edvmr1p0 ~]$ impdp oe/WElcome123##@orclpdb1 schemas=oe
directory=dp_for_oe dumpfile=expoe.dmp sqlfile=oe_SQL
...
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/PRE_SCHEMA/PROCACT_SCHEMA/LOGREP
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
Job "OE"."SYS_SQL_FILE_SCHEMA_01" successfully completed at Thu
Mar 7 00:07:14 2024 elapsed 0 00:00:05

```

```
[oracle@edvmr1p0 ~]$
```

- Review the oe_SQL.sql script. When you execute the import, all DDL statements are read from the dump file.

```

[oracle@edvmr1p0 ~]$ 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:87985C5A76D61CA13FBC93B31E46D8067792F67B0DBEF2517760028A61CE;T
:7F949E004F35C8DA3374A7F968D92C9463043E61F5A0C3612543054AB92D3D4F
94BEC9AC3BFBCF2BEF599A4FFBFED6692D12082E7D75F6643CF29D220CD38BA9D
E401D6D86977B453FA6C285A7B4E07F'
    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
-- new object type path:
SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA/LOGREP

BEGIN
sys.dbms_logrep_imp.instantiate_schema(schema_name=>SYS_CONTEXT(
'USERENV','CURRENT_SCHEMA'), export_db_name=>'ORCLPDB1',
inst_scn=>'14861727');
COMMIT;
END;
/

```

```
-- new object type path: SCHEMA_EXPORT/SEQUENCE/SEQUENCE
CREATE SEQUENCE "OE"."ORDERS_SEQ" MINVALUE 1 MAXVALUE
999999999999 INCREMENT BY 1 START WITH 10 CACHE 20 NOORDER
NOCYCLE NOKEEP NOSCALE GLOBAL ;
-- new object type path: SCHEMA_EXPORT/TABLE(TABLE)
CREATE TABLE "OE"."ORDER_ITEMS"
(
    "ORDER_ID" NUMBER(12,0),
    "LINE_ITEM_ID" NUMBER(3,0),
    "PRODUCT_ID" NUMBER(6,0),
    "UNIT_PRICE" NUMBER(8,2),
    "QUANTITY" NUMBER(8,0)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "TBS_APP2" ;
CREATE TABLE "OE"."ORDERS"
(
    "ORDER_ID" NUMBER(12,0),
    "ORDER_DATE" TIMESTAMP (6) WITH LOCAL TIME ZONE,
    "ORDER_MODE" VARCHAR2(8 BYTE),
    "CUSTOMER_ID" NUMBER(6,0),
    "ORDER_STATUS" NUMBER(2,0),
    "ORDER_TOTAL" NUMBER(12,2),
    "SALES REP_ID" NUMBER(6,0),
    "PROMOTION_ID" NUMBER(6,0)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "TBS_APP" ;
-- new object type path: SCHEMA_EXPORT/TABLE/INDEX/INDEX
CREATE INDEX "OE"."I_ORDER_ITEMS" ON "OE"."ORDER_ITEMS"
("ORDER_ID")
PCTFREE 10 INITTRANS 2 MAXTRANS 255
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS
2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
TABLESPACE "TBS_APP" ;
-- new object type path:
SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
ALTER TABLE "OE"."ORDERS" ADD PRIMARY KEY ("ORDER_ID")
USING INDEX PCTFREE 10 INITTRANS 2 MAXTRANS 255
```

```
STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS  
2147483645  
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1  
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE  
DEFAULT)  
TABLESPACE "TBS_APP" ENABLE;  
-- new object type path:  
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS  
-- new object type path:  
SCHEMA_EXPORT/TABLE/STATISTICS(TABLE_STATISTICS  
-- new object type path: SCHEMA_EXPORT/STATISTICS/MARKER  
-- fixup virtual columns...  
-- done fixup virtual columns  
[oracle@edvmr1p0 ~]$
```

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

10. Start SQL*Plus and connect to ORCLPDB2 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1  
...  
SQL>
```

11. 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  
Help: https://docs.oracle.com/error-help/db/ora-01918/  
SQL>
```

12. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

13. Use Data Pump to import the oe schema. Use the REMAP_SCHEMA parameter to import the entire oe schema into a new oetest schema in ORCLPDB2.

Note: You will get an error message stating that the directory name for DP_FOR_OE is invalid.

```
[oracle@edvmr1p0 ~]$ impdp system/WELCOME123##@orclpdb2  
remap_schema=oe:oetest directory=dp_for_oe dumpfile=expoe.dmp  
  
Import: Release 23.0.0.0.0 - Limited Availability on Thu Mar 7  
00:19:50 2024  
Version 23.4.0.23.11  
  
Copyright (c) 1982, 2023, Oracle and/or its affiliates. All  
rights reserved.  
  
Connected to: Oracle Database 23c Enterprise Edition Release  
23.0.0.0.0 - Limited Availability  
ORA-39002: invalid operation  
ORA-39070: Unable to open the log file.  
ORA-39087: Directory name DP_FOR_OE is invalid.  
[oracle@edvmr1p0 ~]$
```

Question: Why did you receive an error message that the DP_FOR_OE directory does not exist when you created that directory in a previous step?

Answer: You created the directory in ORCLPDB1, not in ORCLPDB2.

14. Connect to ORCLPDB2 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb2  
...  
SQL>
```

15. Create the DP_FOR_OE directory in ORCLPDB2.

```
SQL> CREATE DIRECTORY dp_for_oe AS  
'/u01/app/oracle/admin/orclcdb/dpdump';  
  
Directory created.  
  
SQL>
```

16. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

17. Retry the import operation.

```
[oracle@edvmr1p0 ~]$ impdp system/WELCOME123##@orclpdb2
REMAP_SCHEMA=oe:oetest DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp

Import: Release 23.0.0.0.0 - Limited Availability on Thu Mar 7
00:24:45 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, Oracle and/or its affiliates. All
rights reserved.

Connected to: Oracle Database 23c Enterprise Edition Release
23.0.0.0.0 - Limited Availability
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/LOGREP
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
ORA-39083: Object type TABLE:"OETEST"."ORDER_ITEMS" failed to
create with error:
ORA-00959: tablespace 'TBS_APP2' does not exist

Failing sql is:
CREATE TABLE "OETEST"."ORDER_ITEMS" ("ORDER_ID" NUMBER(12,0),
"LINE_ITEM_ID" NUMBER(3,0), "PRODUCT_ID" NUMBER(6,0),
"UNIT_PRICE" NUMBER(8,2), "QUANTITY" NUMBER(8,0)) SEGMENT
CREATION IMMEDIATE PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS
1 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT) TABLESPACE "TBS_APP2"

Processing object type SCHEMA_EXPORT/TABLE(TABLE_
DATA
. . imported "OETEST"."ORDERS" 12.92 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
Job "SYSTEM"."SYS_IMPORT_FULL_01" completed with 2 error(s) at
Thu Mar 7 00:25:24 2024 elapsed 0 00:00:37
```

```
[oracle@edvmr1p0 ~]$
```

Question: Did the import complete successfully?

Answer: Not completely. Data Pump imported only the objects that it could process without any error.

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.

18. Create the TBS_APP2 tablespace in ORCLPDB2.

- a. Start SQL*Plus and connect to ORCLPDB2 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb2
...
SQL>
```

- b. Issue the CREATE TABLESPACE command to create the TBS_APP2 tablespace in ORCLPDB2.

```
SQL> CREATE TABLESPACE tbs_app2 DATAFILE
  '/u01/app/oracle/oradata/ORCLCDB/orclpdb2/tbs_app02.dbf' SIZE
  100m AUTOEXTEND ON NEXT 25m MAXSIZE 500m;

Tablespace created.

SQL>
```

- c. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

19. Retry the import operation.

```
[oracle@edvmr1p0 ~]$ impdp system/WELCOME123##@orclpdb2
  remap_schema=oe:oetest directory=dp_for_oe dumpfile=expoe.dmp
```

```
Import: Release 23.0.0.0.0 - Limited Availability on Thu Mar 7
00:57:23 2024
Version 23.4.0.23.11

Copyright (c) 1982, 2023, Oracle and/or its affiliates. All
rights reserved.

Connected to: Oracle Database 23c Enterprise Edition Release
23.0.0.0.0 - Limited Availability
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/LOGREP
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" 21.13 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
Job "SYSTEM"."SYS_IMPORT_FULL_01" completed with 3 error(s) at
Thu Mar 7 00:57:33 2024 elapsed 0 00:00:08

[oracle@edvmr1p0 ~]$
```

Question: Are the errors true errors?

Answer: The errors are normal errors stating that objects exist. They were created during the previous import operation.

Verify the OTEST Schema in ORCLPDB2:

Verify that the new oetest schema exists in ORCLPDB2.

20. Start SQL*Plus and connect to ORCLPDB2 as the oetest user.

```
[oracle@edvmr1p0 ~]$ sqlplus oetest/WELCOME123##@orclpdb2  
...  
SQL>
```

21. View the list of tables to which the OTEST user has access.

```
SQL> COL table_name FORMAT A15  
SQL> SELECT table_name FROM user_tables;  
  
TABLE_NAME  
-----  
ORDERS  
ORDER_ITEMS  
  
SQL>
```

22. 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>
```

23. List the indexes to which the oetest user has access.

```
SQL> COL index_name FORMAT A20  
SQL> SELECT index_name FROM user_indexes;  
  
INDEX_NAME  
-----  
SYS_C009375  
I_ORDER_ITEMS  
  
SQL>
```

24. List the sequences to which the oetest user has access.

```
SQL> COL sequence_name FORMAT A20
SQL> SELECT sequence_name FROM user_sequences;

SEQUENCE_NAME
-----
ORDERS_SEQ

SQL>
```

25. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~] $
```

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:

26. Start SQL*Plus and connect to ORCLPDB2 as the system user.

```
[oracle@edvmr1p0 ~] $ sqlplus system/WELCOME123##@orclpdb2
...
SQL>
```

27. Create a database link in the destination PDB (ORCLPDB2) that will connect to the source PDB (ORCLPDB1).

```
SQL> CREATE DATABASE LINK link_orclpdb1 CONNECT TO system
  IDENTIFIED BY WELCOME123## USING 'ORCLPDB1';

Database link created.

SQL>
```

28. Drop the target user created in the previous import operation.

```
SQL> DROP USER oetest CASCADE;

User dropped.

SQL>
```

29. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

30. Invoke Data Pump Import and use the NETWORK_LINK parameter to initiate an import via a database link.

```
[oracle@edvmr1p0 ~]$ impdp system/WELCOME123##@orclpdb2
SCHEMAS=oe REMAP_SCHEMA=oe:oetest NETWORK_LINK=link_orclpdb1
...
Starting "SYSTEM"."SYS_IMPORT_SCHEMA_01":
system/********@orclpdb2 SCHEMAS=oe REMAP_SCHEMA=oe:oetest
NETWORK_LINK=link_orclpdb1
Estimate in progress using BLOCKS method...
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Total estimation using BLOCKS method: 128 KB
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
. . imported "OETEST"."ORDERS" 105
rows
. . imported "OETEST"."ORDER_ITEMS" 665
rows
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type
SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type
SCHEMA_EXPORT/TABLE/STATISTICS(TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER
```

31. Verify that the OE schema was imported as OETEST into ORCLPDB2.

- a. Start SQL*Plus and connect to ORCLPDB2 as the oetest user.

```
[oracle@edvmr1p0 ~]$ sqlplus oetest/WElcome123##@orclpdb2
...
SQL>
```

- b. View the list of tables to which the oetest user has access.

```
SQL> COL table_name FORMAT A20
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
-----
ORDERS
ORDER_ITEMS

SQL>
```

- c. Query the number of rows in the ORDER_ITEMS table. The table has 665 rows.

```
SQL> SELECT count(*) FROM order_items;

COUNT (*)
-----
665

SQL>
```

- d. Exit SQL*Plus.

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

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 24-2: 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

The practice steps indicate when to point to the pluggable database `orclpdb1` or `orclpdb2`.

Tasks

1. Prepare for this practice by opening a terminal window and 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

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_LoadTrans/Trans_Tblspc.sh
The Oracle base remains unchanged with value /u01/app/oracle
Setup done.
[oracle@edvmr1p0 ~]$
```

2. Start a SQL*Plus session and verify the prerequisites for transporting a tablespace across platforms.

- a. Log in as the `sys` user and verify that the source database is in read/write mode.

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
...
SQL> COL name FORMAT A10
SQL> COL open_mode FORMAT A15
SQL> SELECT name, log_mode, open_mode, current_scn FROM
v$database;

NAME          LOG_MODE        OPEN_MODE        CURRENT_SCN
-----        -----          -----          -----
ORCLCDB      NOARCHIVELOG  READ WRITE        14908379

SQL>
```

- b. For performing cross-platform tablespace transport, you must know the exact name of the destination platform to which you are transporting data. Query V\$TRANSPORTABLE_PLATFORM to view the Linux-based platforms by using the query shown in the code box. In the course practice environment, the Linux x86 64-bit platform is used.

```

SQL> COL platform_name FORMAT A30
SQL> SELECT platform_id, platform_name, endian_format
      FROM v$transportable_platform
     WHERE upper(platform_name) LIKE '%LINUX%';

PLATFROM_ID PLATFROM_NAME          ENDIAN_FORMAT
----- -----
    10 Linux IA (32-bit)           Little
    11 Linux IA (64-bit)           Little
     9 IBM zSeries Based Linux    Big
    13 Linux x86 64-bit           Little
    18 IBM Power Based Linux      Big
    22 Linux OS (S64)              Big
    23 Linux OS (AARCH64)         Little

7 rows selected.

SQL>

```

- c. Set orclpdb1 as the current container and make the BARTBS tablespace read only. This is required for the export of the tablespace metadata. Then exit SQL*Plus.

```

SQL> ALTER SESSION SET container=ORCLPDB1;

Session altered.

SQL> ALTER TABLESPACE bartbs READ ONLY;

Tablespace altered.

SQL> EXIT
...
[oracle@edvmr1p0 ~]$

```

- In the same window, start an RMAN session and connect to your `orclpdb1` source database as the target instance.

```
[oracle@edvmr1p0 ~]$ rman target "'sys/WElcome123##@orclpdb1 as sysdba'"
...
connected to target database: ORCLCDB:ORCLPDB1 (DBID=1628325118)

RMAN>
```

- 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;
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 07-MAR-24
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=53 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_aApB":
EXPDP> Processing object type
TRANSPORTABLE_EXPORT/STATISTICS/TABLE_STATISTICS
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_aApB"
successfully loaded/unloaded
EXPDP>
*****
EXPDP> Dump file set for SYS.TRANSPORT_EXP_ORCLCDB_aApB is:
EXPDP>
/u01/app/oracle/product/23.4.0/dbhome_1/dbs/backup_tts_ORCLCDB_99
435.dmp
EXPDP>
*****
EXPDP> Datafiles required for transportable tablespace BARTBS:
EXPDP> /u01/app/backup/ORCLCDB/orclpdb1/bartbs.dbf
EXPDP> Job "SYS"."TRANSPORT_EXP_ORCLCDB_aApB" successfully
completed at Thu Mar 7 01:42:17 2024 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=00064
name=/u01/app/backup/ORCLCDB/orclpdb1/bartbs.dbf
channel ORA_DISK_1: starting piece 1 at 07-MAR-24
channel ORA_DISK_1: finished piece 1 at 07-MAR-24
piece handle=/u01/app/backup/test.bck tag=TAG20240307T014134
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/23.4.0/dbhome_1/dbs/backup_tts_ORCLC
DB_99435.dmp
channel ORA_DISK_1: starting piece 1 at 07-MAR-24
channel ORA_DISK_1: finished piece 1 at 07-MAR-24
piece handle=/u01/app/backup/test.dmp tag=TAG20240307T014134
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:02
Finished backup at 07-MAR-24

```

```
RMAN>
```

5. Enable read/write operations on the BARTBS tablespace. Then exit RMAN.

```

RMAN> ALTER TABLESPACE bartbs READ WRITE;
ALTER TABLESPACE bartbs READ WRITE;
Statement processed

RMAN> EXIT
EXIT

Recovery Manager complete.
[oracle@edvmr1p0 ~]$
```

Note: Normally, after you disconnect from the source database, you move the backup sets and the Data Pump export dump file to the destination host by using operating system utilities. *In this training example, you do not need to do it because you only have one host available.*

6. To simulate target database, set your environment variables to point to the orclcdb database instance, as the sys user connects to the database by using SQL*Plus, and then set the container to orclpdb2.

```
[oracle@edvmr1p0 ~]$ . oraenv
ORACLE_SID = [orclcdb] ? orclcdb
The Oracle base remains unchanged with value /u01/app/oracle
```

```
[oracle@edvmr1p0 ~]$ sqlplus / as sysdba
...
SQL> ALTER SESSION SET container=ORCLPDB2;

Session altered.

SQL>
```

7. Create the bar user in orclpdb2 and grant the CREATE SESSION privilege to bar. Then exit from SQL*Plus.

```
SQL> CREATE USER bar IDENTIFIED BY WELCOME123##;

User created.

SQL> GRANT create session TO bar;

Grant succeeded.

SQL> EXIT
...
[oracle@edvmr1p0 ~]$
```

8. In RMAN, connect to orclpdb2. Use the RESTORE command with the FOREIGN TABLESPACE clause. The FORMAT clause specifies the file destination. Use the DUMP FILE FROM BACKUPSET clause to restore the metadata from the dump file, which is required to plug the tablespace into the destination database.

```
[oracle@edvmr1p0 ~]$ rman target sys/WELCOME123##@orclpdb2
...
connected to target database: ORCLCDB:ORCLPDB2 (DBID=3828180807)

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';

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 07-MAR-24
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=52 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 64 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/23.4.0/dbhome_1/dbs/backup_tts_ORCLCDB_31
312.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: 00:00:01

Performing import of metadata...
IMPDP> Master table "SYS"."TSPITR_IMP_ORCLCDB_shdr"
successfully loaded/unloaded
    IMPDP> Starting "SYS"."TSPITR_IMP_ORCLCDB_shdr":
    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/POST_INSTANCE/PLUGTS_BLK
    IMPDP> Job "SYS"."TSPITR_IMP_ORCLCDB_shdr" successfully
completed at Thu Mar 7 01:59:29 2024 elapsed 0 00:00:30
Import completed

Finished restore at 07-MAR-24

RMAN>

```

9. Confirm that the BARTBS tablespace exists in your destination database. Then exit RMAN.

RMAN> SELECT tablespace_name, status FROM dba_tablespaces;	
TABLESPACE_NAME	STATUS
-----	-----
SYSTEM	ONLINE
SYSAUX	ONLINE

```
UNDOTBS1          ONLINE
TEMP             ONLINE
USERS            ONLINE
TBS_APP          ONLINE
TBS_APP2         ONLINE
BARTBS           READ ONLY
```

8 rows selected

RMAN> **EXIT**

Recovery Manager complete.
[oracle@edvmr1p0 ~]\$

10. Clean up the practice environment by executing the `Trans_Tblspc_cleanup.sh` script. This script removes the original and the transported tablespace, as well as the backup and dump files. The script saves its output in the `/tmp/cleanup.log` file.

```
[oracle@edvmr1p0 ~]$
$HOME/labs/DBMod_LoadTrans/Trans_Tblspc_cleanup.sh
The Oracle base remains unchanged with value /u01/app/oracle
Cleanup complete.
[oracle@edvmr1p0 ~]$
```

11. Exit all terminals.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 25: Using External Tables to Load and Transport Data

Overview

In these practices, you will query and unload external tables.

For Instructor Use Only.
This document should not be distributed.

Practice 25-1: Querying External Tables

Overview

In this practice, you will query partitioned external tables.

Suppose you have 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.

Tasks

1. Open a new terminal window and 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.

```
[oracle@edvmr1p0 ~]$ $HOME/labs/DBMod_LoadTrans/DP_glogin.sh  
The Oracle base remains unchanged with value /u01/app/oracle  
[oracle@edvmr1p0 ~]$
```

2. Start SQL*Plus and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1  
...  
SQL>
```

3. In ORCLPDB1, create the SH.SALES_EXT_RANGE external table.

- a. 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>
```

- b. Create an `sh` schema for the sales data. 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 WELCOME123##;

User created.

SQL> GRANT create session, create table TO sh;

Grant succeeded.

SQL> GRANT read, write ON DIRECTORY ext_dir TO sh;

Grant succeeded.

SQL> GRANT read, write ON DIRECTORY ext_dir2 TO sh;

Grant succeeded.

SQL>
```

- c. View the `$HOME/labs/DBMod_LoadTrans/external_table.sql` script.

```
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>

```

- d. 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.

```

SQL> @$HOME/labs/DBMod_LoadTrans/external_table.sql

Table created.

SQL>

```

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.

- e. 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        DP_sales_1998.dat

```

4. Determine the number of rows in the external table based on specific criteria.

- a. Determine the number of rows for sales in 1998.

```
SQL> SELECT count(*) FROM sh.sales_ext_range partition  
(year1998);  
  
COUNT (*)  
-----  
357668  
  
SQL>
```

- b. Determine the number of rows for sales in 1999.

```
SQL> SELECT count(*) FROM sh.sales_ext_range partition  
(year1999);  
  
COUNT (*)  
-----  
495890  
  
SQL>
```

- c. Determine the number of rows for sales in both 1998 and 1999.

```
SQL> SELECT count(*) FROM sh.sales_ext_range;  
  
COUNT (*)  
-----  
853558  
  
SQL>
```

- d. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~]$
```

- 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.

Note: 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.

```
[oracle@edvmr1p0 ~]$ wc -l
/home/oracle/labs/DBMod_LoadTrans/DP/DP_sales_1998.dat
357675 /home/oracle/labs/DBMod_LoadTrans/DP/DP_sales_1998.dat
[oracle@edvmr1p0 ~]$ wc -l
/home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_1999.dat
495899 /home/oracle/labs/DBMod_LoadTrans/DP2/DP2_sales_1999.dat
```

- Check the log files to determine the reason for the discrepancy.

- List the log files.

Note: Your log file names will be different.

```
[oracle@edvmr1p0 ~]$ ls -l
/home/oracle/labs/DBMod_LoadTrans/DP/*.log
-rw-r--r-- 1 oracle oinstall 5295 Mar  7 21:05
/home/oracle/labs/DBMod_LoadTrans/DP/sh000_670173.log
-rw-r--r-- 1 oracle oinstall 557 Mar  7 21:05
/home/oracle/labs/DBMod_LoadTrans/DP/sh000_720560.log
-rw-r--r-- 1 oracle oinstall 6849 Mar  7 21:05
/home/oracle/labs/DBMod_LoadTrans/DP/sh001_670175.log
[oracle@edvmr1p0 ~]$
```

- 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 a NULL value whereas the column in the table is set to NOT NULL.

```
[oracle@edvmr1p0 ~]$ 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)
...
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)
...
[oracle@edvmr1p0 ~]$
```

- Start SQL*Plus and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WElcome123##@orclpdb1
...
SQL>
```

- Attempt to create an index on the partition key of the external table to get better query performance. The resulting error indicates that you cannot create an index on an external organized table.

```
SQL> CREATE INDEX sh.i_ext_sales_time_id ON sh.sales_ext_range
  (time_id);
CREATE INDEX sh.i_ext_sales_time_id ON sh.sales_ext_range
  (time_id)
*
ERROR at line 1:
ORA-30657: operation not supported on external organized table
Help: https://docs.oracle.com/error-help/db/ora-30657/
SQL>
```

9. 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>

10. Count the number of sales rows in the `SH.SALES_EXT_RANGE` table for year 2000.

Note: The number of rows read equals 235893.

```
SQL> SELECT count(*) FROM sh.sales_ext_range partition (year2000);

COUNT(*)
-----
235893

SQL>
```

11. 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>
```

12. Perform another check on the data. Query the number of rows that have a `TIME_ID` value that falls within the year 2000.

Note: 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>
```

13. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

14. 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 the 2000 sales and another one for the 2001 sales, and then add another partition for the 2001 sales.

```
[oracle@edvmr1p0 ~] $ 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  
...
```

15. Close the terminal window.

Practice 25-2: Unloading External Tables

Overview

In this practice, you will write the OE.ORDERS table data to a dump file using the external tables.

Tasks

1. Open a terminal window and execute the /home/oracle/labs/DBMod_LoadTrans/setup_lab26-2.sh shell script to set formatting for all columns selected in queries, place both .dat files in DP and DP2 subdirectories, create database directories required to access DP and DP2, and then grant READ WRITE privileges on those directories to system user.

```
[oracle@edvmr1p0 ~]$  
/home/oracle/labs/DBMod_LoadTrans/setup_lab26-2.sh  
...  
[oracle@edvmr1p0 ~]$
```

2. Open a terminal window and start SQL*Plus and connect to ORCLPDB1 as the system user.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WELCOME123##@orclpdb1  
...  
SQL>
```

3. In ORCLPDB1, create an external table called OE.ORDERS_EXT that unloads the rows from OE.ORDERS to an external file called orders.dmp. Later, that file will be read from an external table in ORCLPDB2.

```
SQL> CREATE TABLE oe.orders_ext  
ORGANIZATION EXTERNAL  
(TYPE ORACLE_DATAPUMP  
DEFAULT DIRECTORY ext_dir  
LOCATION ('orders.dmp'))  
AS SELECT * FROM oe.orders;  
  
Table created.  
  
SQL>
```

4. Verify that the external file (`orders.dmp`) is listed in the `/home/oracle/labs/DBMod_LoadTrans/DP` directory. The result indicates that it is listed. Your date will be different from the one shown next.

```
SQL> HOST ls -l /home/oracle/labs/DBMod_LoadTrans/DP/orders.dmp
-rw-r----- 1 oracle oinstall 16384 Oct 22 03:02
/home/oracle/labs/DBMod_LoadTrans/DP/orders.dmp
SQL>
```

5. Determine the number of rows in the `OE.ORDERS_EXT` table.

```
SQL> SELECT count(*) FROM oe.orders_ext;
          COUNT(*)
-----
          105
SQL>
```

6. Connect to `ORCLPDB2` as the `system` user and verify the connection to `ORCLPDB2`.

```
SQL> CONNECT system/WELCOME123##@orclpdb2
Connected.
SQL> SHOW CON_NAME
CON_NAME
-----
ORCLPDB2
SQL>
```

7. Drop user `oe` with the `CASCADE` option and recreate the user to clean up in case the user already exists.

```
SQL> DROP USER oe CASCADE;
User dropped.

SQL> CREATE USER oe IDENTIFIED BY WELCOME123##;

User created.

SQL>
```

8. Create an external directory named `ext_dir`.

```
SQL> CREATE DIRECTORY ext_dir AS  
'/home/oracle/labs/DBMod_LoadTrans/DP/';  
  
Directory created.  
  
SQL>
```

9. Create an external table named `OE.ORDERS_EXT` that loads `orders.dmp`.

```
SQL> CREATE TABLE OE.ORDERS_EXT  
(ORDER_ID NUMBER, ORDER_DATE TIMESTAMP(6) WITH LOCAL TIME ZONE,  
ORDER_MODE VARCHAR2(8), CUSTOMER_ID NUMBER(6), ORDER_STATUS  
NUMBER(2), ORDER_TOTAL NUMBER(12), SALES REP_ID NUMBER(6),  
PROMOTION_ID NUMBER(6)) ORGANIZATION EXTERNAL  
(TYPE ORACLE_LOADER DEFAULT DIRECTORY EXT_DIR  
LOCATION ('orders.dmp'));  
2 3 4  
Table created.  
  
SQL>
```

10. Try to query the entire `OE.ORDERS_EXT` table.

Note: 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  
Help: https://docs.oracle.com/error-help/db/ora-29913/  
  
SQL>
```

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.

11. Re-create the external table with the appropriate access driver.

- a. Drop the OE.ORDERS_EXT table that you just created.

```
SQL> DROP TABLE oe.orders_ext;
```

Table dropped.

SQL>

- b. Create the OE.ORDERS_EXT table again, and this time, specify TYPE ORACLE_DATAPUMP (see line 7 in the following) 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(12), SALES REP_ID
  NUMBER(6), PROMOTION_ID NUMBER(6)) ORGANIZATION EXTERNAL
  (TYPE ORACLE_DATAPUMP DEFAULT DIRECTORY EXT_DIR
  LOCATION ('orders.dmp'));
```

Table created.

SQL>

12. Query the entire OE.ORDERS_EXT table again.

Note: This time, the query returns 105 rows.

```
SQL> COL order_date FORMAT A30
SQL> SELECT * FROM oe.orders_ext;

          ORDER_ID ORDER_DATE                      ORDER_MO CUSTOMER_ID
          ORDER_STATUS ORDER_TOTAL SALES REP_ID PROMOTION_ID
-----  -----
0        5458 16-AUG-07 02.34.12.234359 PM    direct      101
       78280           153
1        5397 19-NOV-07 03.41.54.696211 PM    direct      102
       42283           154
1        5454 02-OCT-07 04.49.34.678340 PM    direct      103
       6653           154
...
...
```

```
5456 07-NOV-06 08.53.25.989889 PM direct      117
0      3878          163
5457 31-OCT-07 10.22.16.162632 PM direct      118
5      21586         159

105 rows selected.

SQL>
```

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

```
SQL> EXIT
...
[oracle@edvmr1p0 ~]$ EXIT
```

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 26: Automated Maintenance Tasks Overview

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 27: Managing Tasks and Windows

Overview

In these practices, you will manage maintenance tasks and windows.

For Instructor Use Only.
This document should not be distributed.

Practice 27-1: Enabling and Disabling Automated Maintenance Tasks

Overview

In this practice, you will disable and then re-enable an automated maintenance task.

Tasks

1. Open a terminal window. Using SQL*Plus, log in to ORCLPDB11 as system user. Determine the names of the automated maintenance tasks and the status of each by querying DBA_AUTOTASK_CLIENT.

```
[oracle@edvmr1p0 ~]$ sqlplus system/WElcome123##@orclpdb1
...
SQL> COL client_name FORMAT A40
SQL> SELECT client_name, status FROM dba_autotask_client;

CLIENT_NAME                      STATUS
-----
sql tuning advisor                ENABLED
auto optimizer stats collection   ENABLED
auto space advisor                ENABLED

SQL>
```

2. 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>
```

3. Query DBA_AUTOTASK_CLIENT again to verify that the task is disabled.

```
SQL> SELECT client_name, status FROM dba_autotask_client;

CLIENT_NAME                      STATUS
```

```
-----  
auto optimizer stats collection          ENABLED  
auto space advisor                      ENABLED  
sql tuning advisor                      DISABLED  
  
SQL>
```

4. 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>
```

5. Query DBA_AUTOTASK_CLIENT to verify that the task is once again enabled.

```
SQL> SELECT client_name, status FROM dba_autotask_client;  
  
CLIENT_NAME                      STATUS  
-----  
sql tuning advisor                ENABLED  
auto optimizer stats collection   ENABLED  
auto space advisor                ENABLED  
  
SQL>
```

6. Exit SQL*Plus.

```
SQL> EXIT  
...  
[oracle@edvmr1p0 ~] $
```

Practice 27-2: Modifying the Duration of a Maintenance Window

Overview

In this practice, you will change the duration of the Sunday maintenance window.

Tasks

1. Open a terminal window. Using SQL*Plus, log in to ORCLPDB11 as sys user with the SYSDBA privilege.

```
[oracle@edvmrlp0 ~]$ sqlplus sys/WELCOME123##@orclpdb1 as sysdba  
...  
SQL>
```

2. Determine the duration of the Sunday maintenance window by querying DBA_AUTOTASK_SCHEDULE.

Note: Your output may be different.

```
SQL> COL window_name FORMAT A15  
SQL> COL start_time FORMAT A35  
SQL> COL 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>
```

3. Reset the Sunday maintenance window to 2 hours.

- a. Use the DBMS_SCHEDULER.DISABLE subprogram to disable the Sunday window before making changes to it.

```
SQL> BEGIN  
  dbms_scheduler.disable(  
    name => 'SUNDAY_WINDOW');  
END;  
/
```

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

```
SQL>
```

- b. Use the DBMS_SCHEDULER.SET_ATTRIBUTE subprogram to increase the Sunday maintenance window by 2 hours.

```
SQL> BEGIN
  dbms_scheduler.set_attribute(
    name => 'SUNDAY_WINDOW',
    attribute => 'DURATION',
    value => numtodsinterval(2, 'hour'));
END;
/
```

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

```
SQL>
```

- c. Use the DBMS_SCHEDULER.ENABLE subprogram to re-enable the Sunday window.

```
SQL> BEGIN
  dbms_scheduler.enable(
    name => 'SUNDAY_WINDOW');
END;
/
```

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

```
SQL>
```

4. Verify that the duration of the Sunday maintenance window has changed to 2 hours by querying DBA_AUTOTASK_SCHEDULE again.

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>
```

5. Exit SQL*Plus.

```
SQL> Exit  
...  
[oracle@edvmr1p0 ~] $
```

6. Exit all terminals.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 28: Database Monitoring and Performance Tuning Overview

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 29: Monitoring Database Performance

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.

Practices for Lesson 30: Analyzing SQL and Optimizing Access Paths

For Instructor Use Only.
This document should not be distributed.

Overview

There are no practices for this lesson.

For Instructor Use Only.
This document should not be distributed.