



Integrated Cloud Applications & Platform Services



Oracle Database 12c R2: Administration Workshop

Student Guide – Volume I
D78846GC30
Edition 3.0 | March 2017 | D99461

Learn more from Oracle University at education.oracle.com



Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. 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. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

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 RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

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

Authors

Jody Glover, Jeff Ferreira, Dominique Jeunot, Donna K. Keesling, Mark Fuller

Technical Contributors and Reviewers

James L. Spiller, Jean-Francois Verrier, Hans Forbrich, Jim Spiller, Randy Urbano, Bert Rich, Sarika Surampudi, Kathy Rich, Mike Fitch

Editors

Raj Kumar, Smita Kommini

Graphic Designers

Maheshwari Krishnamurthy, Seema Bopaiyah

Publishers

Jobi Varghese, Sumesh Koshy

Table of Contents

1. Lesson 0 Getting Started	8
1.1 Course Objectives	9
1.2 Course Agenda - Day 1 and 2	10
1.3 Course Agenda - Day 3 and 4	11
1.4 Course Agenda - Day 5	12
1.5 Lab Environment	13
2. Lesson 1 Exploring Oracle Database Architecture	14
2.1 Objectives for Lesson 1	15
2.2 Introducing Oracle Database	16
2.2.1 Oracle Database 12c Editions	17
2.2.2 Oracle Database Standard Edition	19
2.3 Relational Database Models	20
2.4 Oracle SQL	22
2.5 Oracle Database Server Architecture	24
2.5.1 Database Instance	26
2.5.2 Multitenant Database	28
2.5.3 Logical Storage Structures	30
2.5.4 Database Files	32
2.5.5 Example - User Issuing a SQL Statement	34
2.6 Connecting to Oracle Databases	36
2.7 Oracle Database Tools	38
2.7.1 SQLPlus	40
2.7.2 SQLcl	42
2.7.3 Oracle SQL Developer	43
2.7.4 Database Configuration Assistant	45
2.7.5 Oracle Enterprise Manager Database Express	46
2.7.6 Oracle Enterprise Manager Cloud Control	48
2.8 Oracle-Supplied User Accounts	50
2.9 Querying the Oracle Data Dictionary	52
2.10 Summary for Lesson 1	54
2.11 Practice 1 Overview	55
2.11.1 Practice 1-1 Exploring a CDB	56
2.11.2 Practice 1-2 Exploring a PDB	67
2.11.3 Practice 1-3 Exploring a CDB and PDB with EM Express	71
2.11.4 Practice 1-4 Getting Started with SQLcl	77
3. Lesson 2 Managing Database Instances	82
3.1 Objectives for Lesson 2	83
3.2 Working with Initialization Parameters	84
3.2.1 Initialization Parameters	86
3.2.2 Modifying Initialization Parameters	88
3.2.3 Viewing Initialization Parameters	91
3.3 Starting Up Oracle Databases	93
3.4 Shutting Down Oracle Databases	94
3.5 Opening and Closing PDBs	97
3.6 Working with the Automatic Diagnostic Repository	99
3.6.1 Viewing the Alert Log	100
3.6.2 Using Trace Files	101
3.6.3 Administering the DDL Log File	103
3.7 Querying Dynamic Performance Views	104
3.7.1 Considerations for Dynamic Performance Views	105
3.8 Summary for Lesson 2	106
3.9 Practice 2 Overview	107
3.9.1 Practice 2-1 Creating a PFILE from an SPFILE	108
3.9.2 Practice 2-2 Viewing Initialization Parameters	116
3.9.3 Practice 2-3 Modifying Initialization Parameters	130
3.9.4 Practice 2-4 Modifying an Initialization Parameter with EM Express	139
3.9.5 Practice 2-5 Shutting Down and Starting Up the Oracle Database	142
3.9.6 Practice 2-6 Viewing Diagnostic Information	148
4. Lesson 3 Creating PDBs	164
4.1 Objectives for Lesson 3	165
4.2 Methods and Tools to Create PDBs	166
4.3 Creating PDBs from Seed	167
4.4 Cloning PDBs	169
4.5 Unplugging and Plugging In PDBs	171
4.6 Dropping PDBs	173
4.7 Summary for Lesson 3	175
4.8 Practice 3 Overview	176
4.8.1 Practice 3-1 Creating a PDB from Seed	177
4.8.2 Practice 3-2 Hot Cloning a PDB	185
4.8.3 Practice 3-3 Unplugging and Plugging a PDB	193
4.8.4 Practice 3-4 Dropping a PDB	198
5. Lesson 4 Configuring the Oracle Network Environment	200
5.1 Objectives for Lesson 4	201

5.2 Oracle Net Services	202
5.3 How Listeners Work	204
5.3.1 The Default Listener	206
5.3.2 Tools to Configure Listeners	207
5.4 Configuring Dynamic Service Registration	208
5.5 Configuring Static Service Registration	210
5.6 Configuring Local Naming for Connections	212
5.6.1 Advanced Connection Options	214
5.7 Testing Oracle Net Connectivity with tnsping	216
5.8 Configuring Communication Between Databases	217
5.9 Dedicated Versus Shared Server Configurations	219
5.10 Summary for Lesson 4	222
5.11 Practice 4 Overview	223
5.11.1 Practice 4-1 Exploring the Default Listener	224
5.11.2 Practice 4-2 Creating a Dynamic Listener	232
5.11.3 Practice 4-3 Creating a Static Listener for a PDB	243
5.11.4 Practice 4-4 Creating a Net Service Name	247
5.11.5 Practice 4-5 Creating a Database Link to an External Database	259
6. Lesson 5 Administering User Security	273
6.1 Objectives for Lesson 5	274
6.2 Creating Users	275
6.2.1 User Accounts	277
6.2.2 Oracle-Supplied Administrator Accounts	279
6.3 Granting Privileges	280
6.3.1 System Privileges	282
6.3.2 Object Privileges	285
6.4 Creating and Granting Roles	286
6.4.1 Granting Roles	288
6.4.2 Oracle-Supplied Roles	289
6.4.3 Making Roles More Secure	290
6.5 Revoking Privileges and Roles	292
6.5.1 Revoking System Privileges	293
6.5.2 Revoking Object Privileges	294
6.5.3 Revoking Roles	295
6.6 Creating and Assigning Profiles	296
6.6.1 Assigning Profiles	298
6.6.2 Password Parameters	299
6.6.3 Resource Parameters	301
6.6.4 Oracle-Supplied Password Verification Functions	303
6.7 Authenticating Users	305
6.7.1 Password Authentication	307
6.7.2 Password File Authentication	308
6.7.3 OS Authentication	309
6.8 Assigning Quotas to Users	312
6.9 Applying the Principle of Least Privilege	313
6.10 Summary for Lesson 5	315
6.11 Practice 5 Overview	316
6.11.1 Practice 5-1 Creating Common and Local Users	317
6.11.2 Practice 5-2 Creating a Local User for an Application	324
6.11.3 Practice 5-3 Granting the DBA Role to PDBADMIN	327
6.11.4 Practice 5-4 Creating a Local Profile in EM Express and Locking Accounts	330
6.11.5 Practice 5-5 Creating Local Roles in EM Express	338
6.11.6 Practice 5-6 Creating Local Users in EM Express	345
6.11.7 Practice 5-7 Configuring a Default Role for a User	357
6.11.8 Practice 5-8 Auditing User Activity	360
6.11.9 Practice 5-9 Exploring OS and Password File Authentication	366
7. Lesson 6 Creating and Managing Tablespaces	371
7.1 Objectives for Lesson 6	372
7.2 How Table Data Is Stored	373
7.2.1 Database Block Content	374
7.3 Creating Tablespaces	375
7.4 Altering and Dropping Tablespaces	378
7.5 Viewing Tablespace Information	380
7.6 Implementing Oracle Managed Files	381
7.7 Moving or Renaming Online Data Files	383
7.7.1 Examples of Moving and Renaming Online Data Files	385
7.8 Summary for Lesson 6	386
7.9 Practice 6 Overview	387
7.9.1 Practice 6-1 Viewing Tablespace Information	388
7.9.2 Practice 6-2 Creating a Tablespace	396
8. Lesson 7 Managing Storage Space	410
8.1 Objectives for Lesson 7	411
8.2 Space Management Features	412
8.3 Block Space Management	413
8.4 Row Chaining and Migration	414

8.5 Free Space Management Within Segments	416
8.6 Types of Segments	417
8.7 Allocating Extents	418
8.8 Deferred Segment Creation	419
8.8.1 Controlling Deferred Segment Creation	420
8.8.2 Restrictions and Exceptions	421
8.9 Space-Saving Features	422
8.10 Table Compression Overview	423
8.10.1 Compression for Direct-Path Insert Operations	424
8.10.2 Advanced Row Compression for DML Operations	425
8.10.3 Specifying Table Compression	426
8.10.4 Using the Compression Advisor	427
8.11 Resolving Space Usage Issues	428
8.12 Monitoring Tablespace Space Usage	429
8.13 Reclaiming Space by Shrinking Segments	430
8.13.1 Shrinking Segments	431
8.13.2 Results of a Shrink Operation	432
8.14 Managing Resumable Space Allocation	433
8.14.1 Using Resumable Space Allocation	434
8.14.2 Resuming Suspended Statements	436
8.15 Summary for Lesson 7	438
8.16 Practice 7 Overview	439
8.16.1 Practice 7-1 Managing Tablespace Space	440
8.16.2 Practice 7-2 Using Compression	450
8.16.3 Practice 7-3 Handling Resumable Space Allocation	457
9. Lesson 8 Managing UNDO Data	464
9.1 Objectives for Lesson 8	465
9.2 Undo Data Overview	466
9.3 Transactions and Undo Data	468
9.4 Storing Undo Information	469
9.5 Comparing Undo Data and Redo Data	470
9.6 Managing Undo	471
9.7 Local Undo Mode Versus Shared Undo Mode	472
9.8 Configuring Undo Retention	473
9.9 Categories of Undo	474
9.10 Guaranteeing Undo Retention	475
9.11 Changing an Undo Tablespace to a Fixed Size	476
9.12 Temporary Undo Overview	477
9.13 Temporary Undo Benefits	478
9.14 Enabling Temporary Undo	479
9.15 Monitoring Temporary Undo	480
9.16 Viewing Undo Information	481
9.17 Viewing Undo Activity	482
9.18 Practice 8 Overview	483
9.18.1 Practice 8-1 Managing UNDO Data	484
9.18.2 Practice 8-2 Using Local UNDO	490
10. Lesson 9 Moving Data	498
10.1 Objectives for Lesson 9	499
10.2 Moving Data - General Architecture	500
10.3 Oracle Data Pump Overview	501
10.4 Oracle Data Pump Benefits	502
10.5 Data Pump Export and Import Clients	504
10.6 Data Pump Interfaces and Modes	505
10.7 Data Pump Import Transformations	507
10.8 SQL Loader Overview	508
10.9 Loading Methods	510
10.10 Express Mode	511
10.11 External Tables	513
10.12 External Table Benefits	514
10.13 Summary for Lesson 9	515
10.14 Practice 9 Overview	516
10.14.1 Practice 9-1 Moving Data From One PDB to Another	517
10.14.2 Practice 9-2 Loading Data into a PDB from an External File	531
10.14.3 Practice 9-3 Querying External Tables	545
10.14.4 Practice 9-4 Unloading External Tables	553
11. Lesson 10 Backup and Recovery Concepts	557
11.1 Objectives for Lesson 10	558
11.2 DBA Responsibilities	559
11.3 Categories of Failure	560
11.3.1 Statement Failure	561
11.3.2 User Process Failure	562
11.3.3 Network Failure	563
11.3.4 User Error	564
11.3.5 Instance Failure	565
11.3.6 Media Failure	566

11.4 Understanding Instance Recovery	567
11.4.1 The Checkpoint Process	568
11.4.2 Redo Log Files and Log Writer	569
11.4.3 Automatic Instance or Crash Recovery	570
11.4.4 Phases of Instance Recovery	571
11.4.5 Tuning Instance Recovery	572
11.4.6 Using the MTTR Advisor	573
11.5 Understanding Types of Backups	574
11.5.1 Backup Terminology	575
11.5.2 Types of Backups	576
11.5.3 RMAN Backup Types	577
11.6 Comparing Complete and Incomplete Recovery	579
11.6.1 Complete Recovery Process	580
11.6.2 Point-in-Time Recovery Process	581
11.7 Oracle Data Protection Solutions	583
11.8 Flashback Technology	584
11.9 Summary for Lesson 10	585
11.10 Practice 10 Overview	586
11.10.1 Practice 10-1 Configuring Your Database for Recovery	587
11.10.2 Practice 10-2 Backing up the Control File	601
11.10.3 Practice 10-3 Configuring Automatic Backups of the Control File and SPFILE	606
11.10.4 Practice 10-4 Creating a Whole Database Backup	610
11.10.5 Practice 10-5 Creating a Partial Database Backup	627
11.10.6 Practice 10-6 Recovering From an Essential Data File Loss	633
11.10.7 Practice 10-7 Recovering From an Application Data File Loss	647
12. Lesson 11 Monitoring and Tuning Database Performance	661
12.1 Objectives for Lesson 11	662
12.2 Performance Management Activities	663
12.3 Performance Planning Considerations	664
12.4 Database Maintenance	666
12.5 Automatic Workload Repository	667
12.6 Automatic Database Diagnostic Monitor	668
12.7 Performance Monitoring	669
12.7.1 Monitoring Sessions	670
12.7.2 Monitoring Services	671
12.7.3 Monitoring Wait Events	672
12.7.4 Viewing Instance Statistics	673
12.8 Performance Tuning Methodology	675
12.9 Database Server Statistics and Metrics	676
12.10 Managing Memory Components	677
12.10.1 Automatic Memory Management	679
12.10.2 Automatic Shared Memory Management	681
12.10.3 Managing the SGA for PDBs	683
12.10.4 Managing the Program Global Area	684
12.10.5 Managing the PGA for a PDB	686
12.11 Summary for Lesson 11	687
12.12 Practice 11 Overview	688
12.12.1 Practice 11-1 Managing Performance in EM Express	689
12.12.2 Practice 11-2 Resolving Lock Issues	696
13. Lesson 12 Tuning SQL	702
13.1 Objectives for Lesson 12	703
13.2 SQL Tuning Process	704
13.3 Oracle Optimizer	705
13.4 Optimizer Statistics	706
13.4.1 Optimizer Statistics Collection	707
13.4.2 Setting Optimizer Statistics Preferences	709
13.4.3 Optimizer Statistics Advisor	711
13.5 SQL Plan Directives	713
13.6 Adaptive Execution Plans	714
13.7 SQL Tuning Advisor	716
13.8 SQL Access Advisor	718
13.9 SQL Performance Analyzer	719
13.10 Summary for Lesson 12	721
13.11 Practice 12 Overview	722
13.11.1 Practice 12-1 Using SQL Tuning Advisor	723
13.11.2 Practice 12-2 Using Optimizer Statistics Advisor	733
14. Lesson 13 Oracle Database Resource Manager	743
14.1 Objectives for Lesson 13	744
14.2 Oracle Database Resource Manager Overview	745
14.3 Resource Manager Elements	746
14.4 Using Resource Manager to Allocate Resources	747
14.5 Creating a Simple Resource Plan	749
14.6 Creating a Complex Resource Plan	750
14.7 Using the Active Session Pool Feature	752
14.8 Limiting CPU Utilization at the Database Level	753

14.9 Limiting CPU Utilization at the Server Level	754
14.10 Viewing Resource Manager Information	756
14.11 Limiting PGA Memory Allocated to Each Session	757
14.12 Creating Directives for PDB Performance Profiles	758
14.13 Summary for Lesson 13	759
14.14 Practice 13 Overview	760
14.14.1 Practice 13-1 Dispatching Resources Amongst PDBs by Using Resource Manager	761
14.14.2 Practice 13-2 Avoiding Excessive PGA Memory Usage	768
15. Lesson 14 Enterprise Manager Cloud Control	773
15.1 Objectives for Lesson 14	774
15.2 Controlling the Enterprise Manager Cloud Control Framework	775
15.3 Starting the Enterprise Manager Cloud Control Framework	776
15.4 Stopping the Enterprise Manager Cloud Control Framework	778
15.5 Summary for Lesson 14	779
15.6 Practice 14 Overview	780
15.6.1 Practice 14-1 Registering a Database with Enterprise Manager Cloud Control	781
15.6.2 Practice 14-2 Using ADDM	791
16. Appendices	796
16.1 Appendix - Product-Specific Credentials	797
16.2 Appendix - Abbreviations	799
16.3 Appendix - Processes	800
16.4 Appendix - SGA Components	807

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Getting Started



ORACLE®

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Course Objectives

Unauthorized reproduction or distribution prohibited
Copyright © 2017, Oracle and/or its affiliates.

After completing this course, you should be able to:

- Describe Oracle Database architecture
- Configure the database to support your applications
- Manage database security and implement auditing
- Implement basic backup and recovery procedures
- Move data between databases and files
- Employ basic monitoring procedures and manage performance
- Manage resources and automate tasks
- Create database deployments in Database Cloud Service (DBCS)
- Manage database deployments in DBCS



ORACLE®

Course Agenda - Day 1 and 2

- Day 1
 - Lesson 1 Exploring Oracle Database Architecture
 - Lesson 1C Introduction to Oracle Database Cloud Service
 - Lesson 2C Creating DBCS Database Deployments
 - Lesson 2 Managing Database Instances
- Day 2
 - Lesson 3 Creating PDBs
 - Lesson 6C Creating Master Encryption Keys for PDBs
 - Lesson 4 Configuring the Oracle Network Environment
 - Lesson 3C Managing DBCS Database Deployments

ORACLE

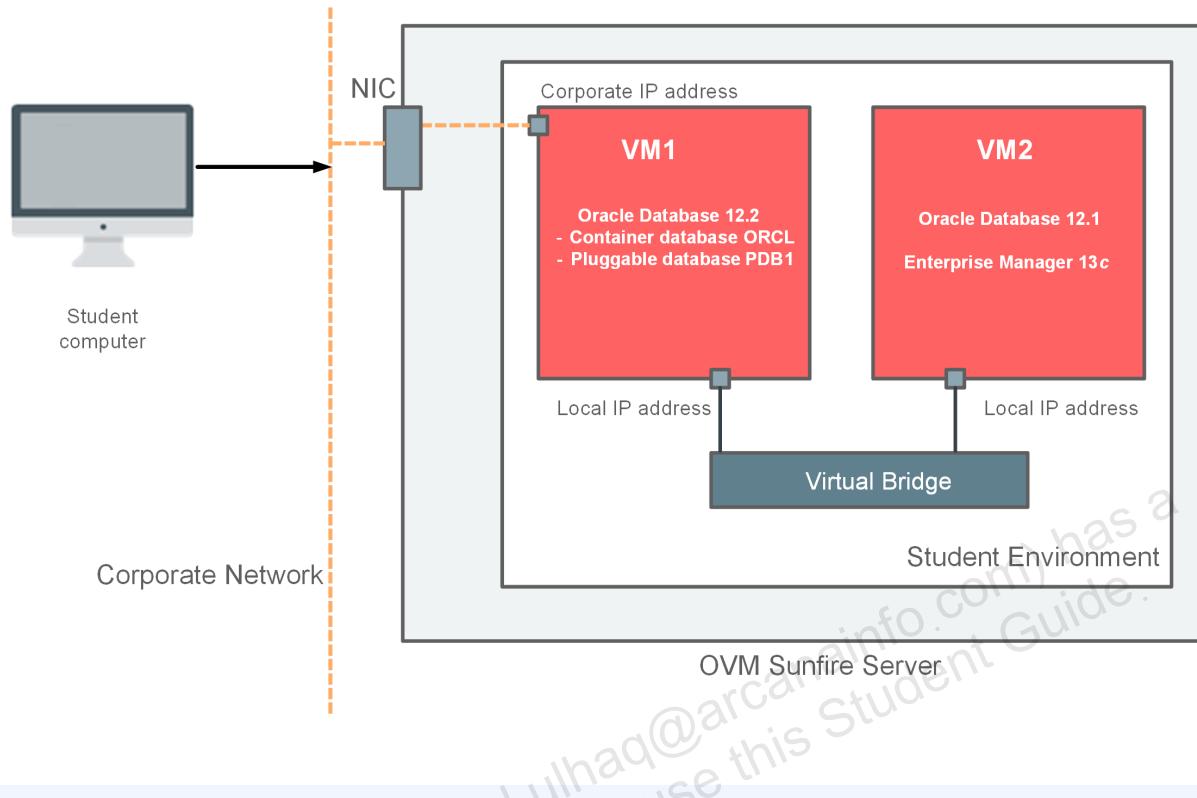
Course Agenda - Day 3 and 4

- Day 3
 - Lesson 5C Patching DBCS Database Deployments
 - Lesson 5 Administering User Security
 - Lesson 6 Creating and Managing Tablespaces
 - Lesson 7C Tablespace Encryption by Default
 - Lesson 7 Managing Storage Space
- Day 4
 - Lesson 8 Managing UNDO Data
 - Lesson 9 Moving Data
 - Lesson 10 Backup and Recovery Concepts
 - Lesson 4C Backing Up and Restoring DBCS Database Deployments

Course Agenda - Day 5

- Day 5
 - Lesson 11 Monitoring and Tuning Database Performance
 - Lesson 12 Tuning SQL
 - Lesson 13 Oracle Database Resource Manager
 - Lesson 14 Enterprise Manager Cloud Control

Lab Environment



ORACLE

VM1 and VM2

Each student has two virtual machines (VM1 and VM2) to use during class, as shown in the diagram above:

- On VM1, Oracle Database 12.2 is installed with one container database named ORCL and one pluggable database named PDB1. This container database (CDB) has been set up for you in advance so that you can explore an existing configuration. The host name for VM1 is 12cr2db.
- On VM2, Oracle Database 12.1 (non-container database) and Enterprise Manager 13c are installed. Oracle Database 12.1 is required for Enterprise Manager 13c. You'll use this virtual machine to practice managing Oracle Database 12.2 with Oracle Enterprise Manager Cloud Control. The host name for VM2 is em13c.

See Appendix - Product-Specific Credentials for a list of credentials used in the practices for this course.

1

Exploring Oracle Database Architecture



ORACLE

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Objectives for Lesson 1

After completing this lesson, you should be able to:

- Describe Oracle Database and its editions
- Explain the general structure of a relational database model
- Describe SQL and PL/SQL
- Describe the architecture of a multitenant Oracle Database
- Connect to an Oracle Database by using OS authentication and Easy Connect Syntax
- Describe the main Oracle Database tools
- Describe the Oracle-supplied user accounts for Oracle Database
- Query the data dictionary in an Oracle Database



ORACLE®

Introducing Oracle Database

- Oracle provides cloud and on-premises database offerings.
- The purpose of an Oracle Database is to store, organize, and retrieve data for your applications.
- You can run Oracle Database in your environment (on-premises) or use Oracle Database in Oracle's environment (cloud).
- Your applications are the user interfaces to the database.
 - Users can retrieve data in the database to create reports.
 - Users can update data in the database.



Oracle Database 12c Editions

Oracle Database is available in the following editions, each suitable for different development and deployment scenarios.

- Oracle Database PE
- Oracle Database SE2
- Oracle Database EE
- Oracle Database XE



Oracle Database Personal Edition

Oracle Database Personal Edition supports single-user development and deployment environments that require full compatibility with Oracle Database Standard Edition One, Oracle Database Standard Edition, and Oracle Database Enterprise Edition.

Personal Edition includes all of the components that are included with Enterprise Edition, as well as all of the options that are available with Enterprise Edition. Personal Edition is available on Windows platforms only. The Management Packs are not included in Personal Edition.

Note: You cannot use the Oracle Real Application Clusters option with Personal Edition.

Oracle Database Standard Edition2

Before SE2, there is SE (Standard Edition) and SE1 (Standard Edition1).

- Oracle Database Standard Edition delivers the unprecedented ease of use, power, and performance of Standard Edition One, with support for larger machines and clustering of services with Oracle Real Application Clusters (Oracle RAC). Oracle RAC is not included in the Standard Edition of releases before Oracle Database 10g, nor is it an available option with those earlier releases.

- Oracle Database Standard Edition One delivers unprecedented ease of use, power, and performance for workgroup, department-level, and Web applications. From single-server environments for small business to highly distributed branch environments, Oracle Database Standard Edition One includes all the facilities necessary to build business-critical applications.

SE2 starts with Oracle Database 12c Release 1 (12.1.0.2).

SE2 supports Oracle RAC.

Oracle Database Enterprise Edition

Oracle Database Enterprise Edition provides the performance, availability, scalability, and security required for mission-critical applications such as high-volume online transaction processing (OLTP) applications, query-intensive data warehouses, and demanding Internet applications. Oracle Database Enterprise Edition contains all of the components of Oracle Database, and can be further enhanced with the purchase of the options and packs.

- Oracle Database Options
 - All the Oracle Database options can be purchased with Oracle Database Enterprise Edition.
 - You may not use the options, packs, or products without separately purchased licenses. The fact that these options, packs, or products may be included in product CDs or downloads or described in documentation that you receive, does not authorize you to use them without purchasing appropriate licenses.
- Oracle Management Packs
 - The management packs can be purchased only with Enterprise Edition.
 - The features in these packs are accessible through Oracle Enterprise Manager Database Control, Oracle Enterprise Manager Grid Control, and APIs provided with Oracle Database software.

Oracle Database Express Edition

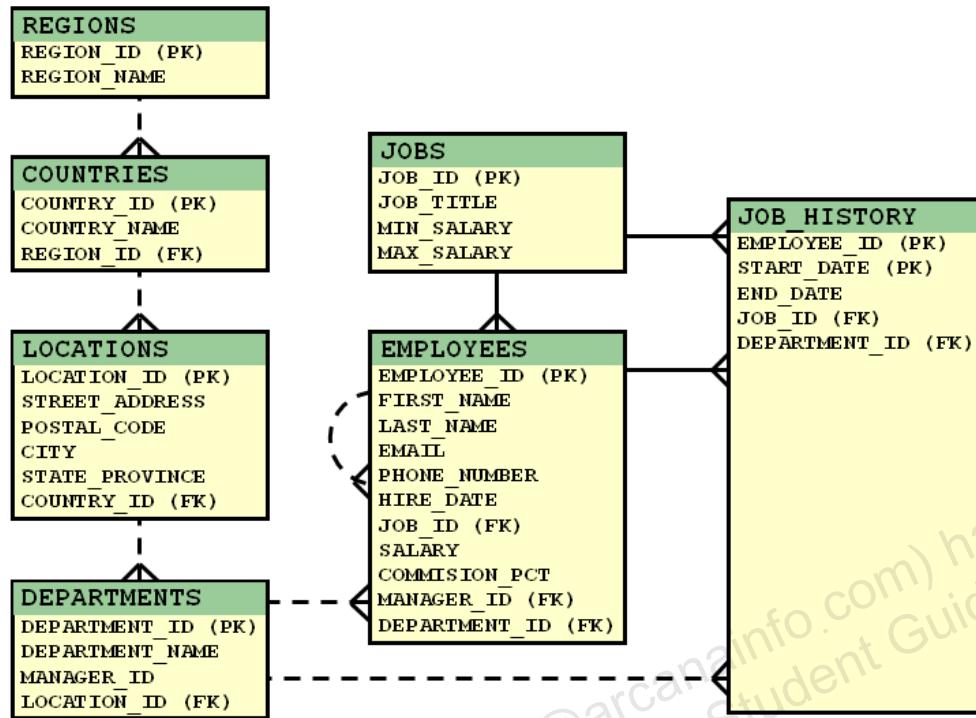
Oracle Database Express Edition (Oracle Database XE) is an entry-level edition of Oracle Database that is quick to download, simple to install and manage, and is free to develop, deploy, and distribute. Oracle Database XE makes it easy to upgrade to the other editions of Oracle without costly and complex migrations. Oracle Database XE can be installed on any size machine with any number of CPUs, stores up to 4 GB of user data, using up to 1 GB of memory, and using only one CPU on the host machine. An online forum provides the support.

Oracle Database Standard Edition

- From Oracle Database 12c Release 1 (12.1.0.2), SE2 replaces SE and SE1.
- SE2 supports Oracle RAC.
- SE2 (and SE, SE1) has the same main kernel.
- It supports single tenant but lacks the following features, options and tools:
 - No parallel execution
 - No Data Guard
 - No Cloud Control
 - No management packs

ORACLE

Relational Database Models



ORACLE

About Relational Database Models

Oracle Database uses a relational database model, which organizes and presents the physical data as *logical* structures, such as tables (with columns and rows). These logical structures make the data understandable. A *schema* is a collection of logical structures that are owned by a database user, and can include tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, schemas include everything that your application creates in the database.

HR Schema

The HR schema, which is shown above and is used throughout this course, groups seven related tables and was created by the HR user. A primary key (identified by PK above) is a unique identifier for a row in a table. A foreign key (identified by FK above) references a primary key. It uses the same value as the primary key. Together, primary keys and foreign keys are used to ensure data integrity. For example, in the JOBS table above, each row uniquely identifies a job. Each job is given a job ID, and this ID is the primary key. Likewise, the EMPLOYEES table uniquely identifies each employee with an employee ID. The employee ID is also a primary key. The EMPLOYEES table contains a foreign key named JOB_ID, which references the JOB_ID primary key in the JOBS table. Solid lines in the diagram represent mandatory foreign key constraints and dashed lines represent optional foreign key constraints.

The following are some principal business rules implemented in the HR schema:

- Each department may be the employer of one or more employees. Each employee may be assigned to only one department.

- Each job must be a job for one or more employees. Each employee must be currently assigned to only one job.
- When an employee changes his or her department or job, a record in the `JOB_HISTORY` table records the start and end dates of the past assignments.
- The `JOB_HISTORY` table records are identified by a composite primary key: the `EMPLOYEE_ID` and the `START_DATE` columns.
- The `EMPLOYEES` table also has a foreign key constraint with itself. This is an implementation of the business rule: each employee may be reporting directly to only one manager. The foreign key is optional because the top employee does not report to another employee.

Oracle SQL

- Oracle SQL is the language you use to perform operations on the data in an Oracle Database. For example, selecting data from a database:

```
SQL> SELECT employee_id, first_name, last_name FROM employees  
      WHERE employee_id=216 ORDER BY 1;
```

- SELECT lists the database columns for which you want to view data.
- FROM lists the tables that contain those database columns.
- WHERE specifies column limits and table joins (this part essentially filters the rows of data).
- ORDER BY specifies the columns by which the results are sorted.
- PL/SQL is a procedural extension to Oracle SQL.
 - It enables you to control the flow of a SQL program, use variables, and write error-handling procedures.

ORACLE

Oracle SQL, DDL, and DML

Oracle SQL is the language you use to perform operations on the data in an Oracle Database. It's an implementation of the ANSI standard language called Structure Query Language (SQL), and provides features that extend beyond standard SQL. For example, you use SQL to create tables, query tables, and modify data in tables. A SQL statement can be thought of as a very simple, but powerful, computer program or instruction. Users specify the result that they want (for example, the names of employees), not how to derive it. A SQL statement is a string of SQL text such as the following:

```
SELECT first_name, last_name FROM employees;
```

Data definition language (DDL) includes statements that define or change a data structure; for example, CREATE TABLE and ALTER INDEX statements.

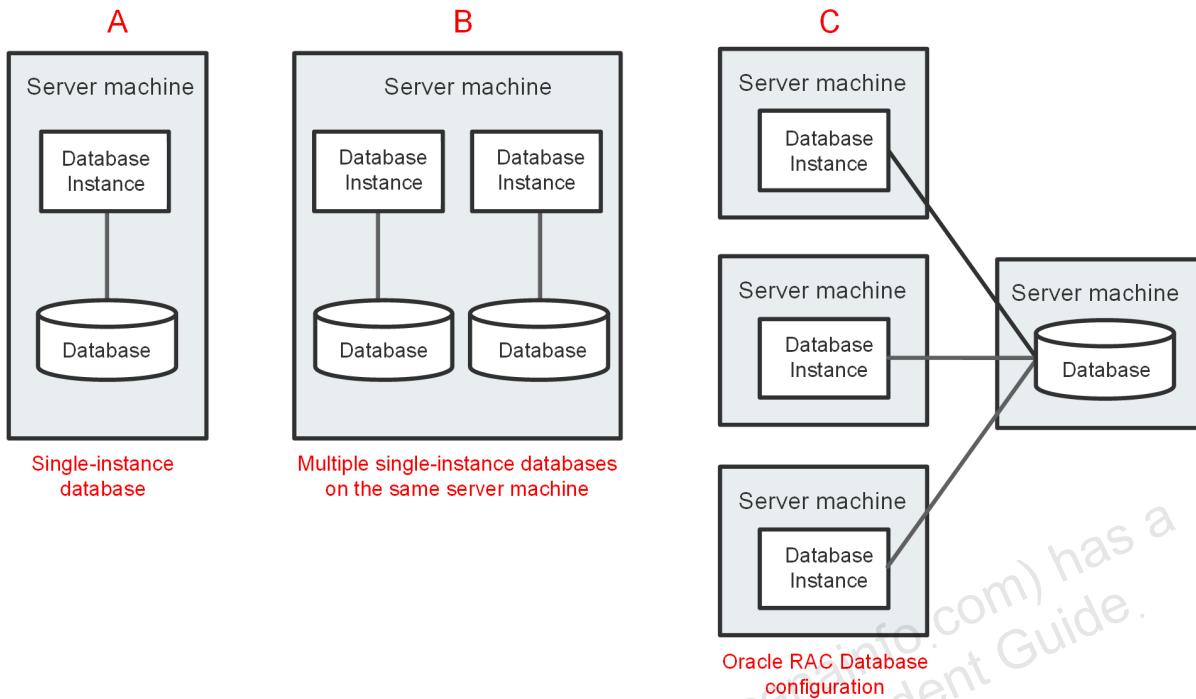
Data manipulation language (DML) includes statements such as SELECT, INSERT, UPDATE, and DELETE.

PL/SQL

PL/SQL is a procedural extension to Oracle SQL. PL/SQL is integrated with Oracle Database, enabling you to use all of the Oracle Database SQL statements, functions, and data types. You can use PL/SQL to control the flow of a SQL program, use variables, and write error-handling procedures.

Oracle Database can also store program units written in Java. A *Java stored procedure* is a Java method published to SQL and stored in the database for general use. You can call existing PL/SQL programs from Java and Java programs from PL/SQL.

Oracle Database Server Architecture



ORACLE®

Configuration Options

An Oracle Database consists of at least one database instance to handle memory and processes and one database, which consists of physical files.

An on-premises Oracle Database has the following configuration options, as illustrated in the diagram above.

- A single-instance database (**A**): This architecture consists of one database instance and one database. A one-to-one relationship exists between the database and a database instance. You will work with this architecture during this course.
- Multiple single-instance databases on the same server machine (**B**): You can install multiple single-instance databases on the same server machine, where there will still be a separate database instance for each database. This configuration is useful for running different versions of Oracle Database on the same machine.
- Oracle Real Application Clusters (Oracle RAC) database (**C**): This architecture has multiple instances usually running on separate server machines, all sharing the same database. The cluster of server machines appear as if they are one server to end users and applications. This configuration is designed for high availability, scalability, and high-end performance.

Whether in a single-instance or Oracle RAC configuration, a database instance is associated with only one database at a time.

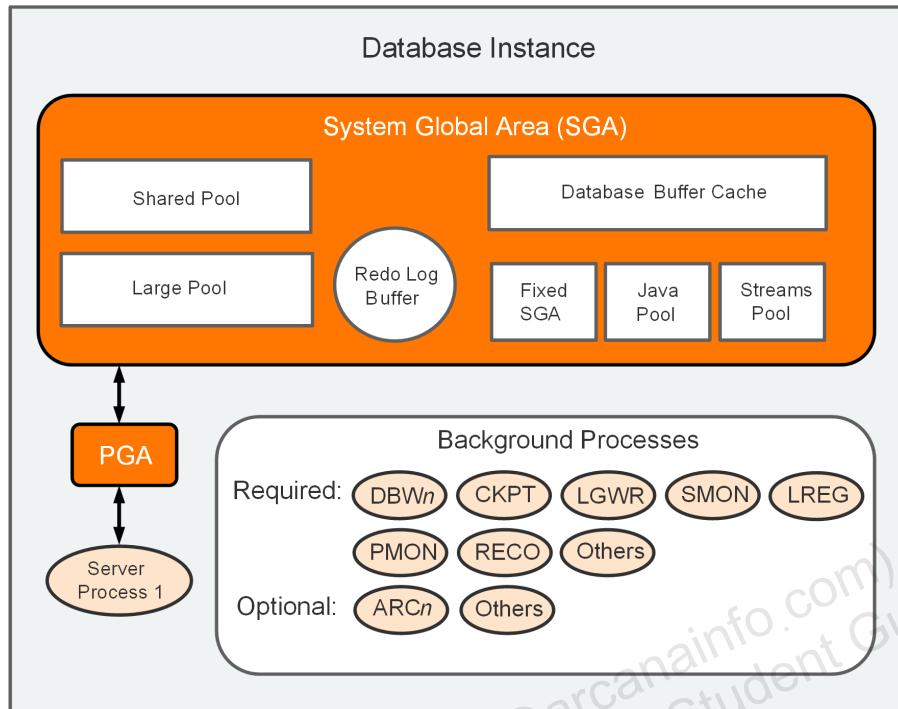
 **View:**

View the following figures in the *Oracle Database Net Services Administrator's Guide*. In Figure 2-1, each database instance (sales and finance) is associated with its own service name (`sales.us.example.com` and `finance.us.example.com`, respectively). Figure 2-2 is an advanced scenario involving Oracle Real Application Clusters (RAC) Database where two database instances are associated with the same database service. Both database instances run on different computers and share a single physical database.

- Figure 2-1 and Figure 2-2 (Click the second Understanding Database Instances link.)

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Database Instance



ORACLE®

A *database instance*, as illustrated above, is a set of memory structures and processes that access and process data from the database files. A database instance exists only in memory.

The main memory structures in a database instance are the following:

- **System Global Area (SGA):** The SGA is a group of shared memory structures that contain data and control information for one database instance. All server and background processes share the SGA. SGA components include a shared pool, database buffer cache, redo log buffer, large pool, java pool, streams pool, and fixed SGA. The In-Memory Area is an optional SGA component that contains the In-Memory Column Store (IM column store). See [Appendix - SGA Components](#) for descriptions of these items.
- **Program Global Area (PGA):** The PGA is a nonshared memory region that contains data and control information exclusively for use by server and background processes. One PGA gets created for each server or background process that starts. The Oracle Database creates server processes to handle the connections to the database on behalf of client programs. A PGA is deallocated when the associated server or background process using it is terminated.

The main process structures in a database instance are the following:

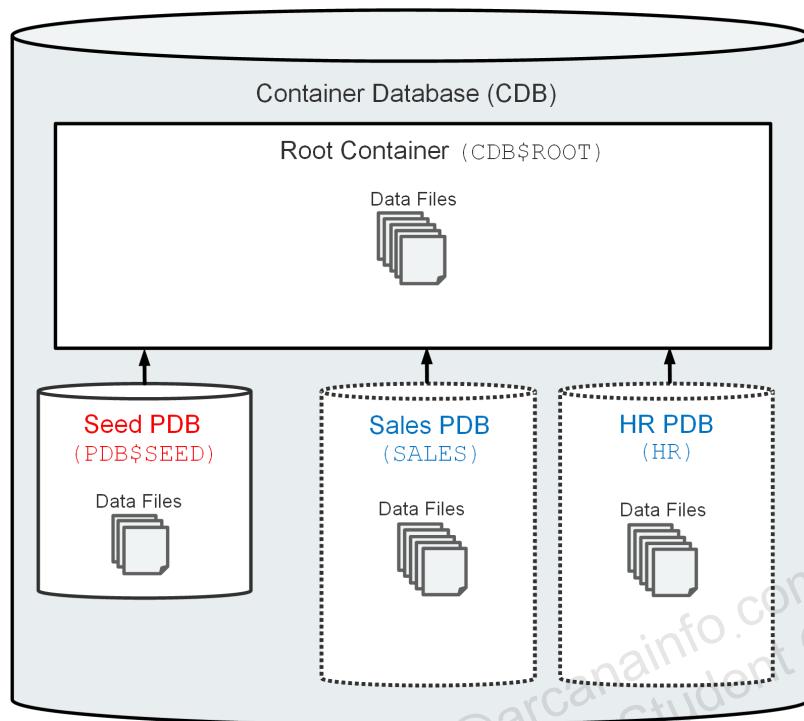
- **Background processes:** These processes manage memory structures, asynchronously perform I/O to write data to a file on a disk, and perform general maintenance tasks. The background processes that are present depend on the features that are being used in the database. When you start a database instance, mandatory background processes automatically start. You can start optional background processes later as required. The following are some common background processes. For descriptions of individual background process, see

Appendix - Processes. For a complete list of background processes, see **Table F-1** in *Oracle Database Reference*.

- Database Writer process (DBWN)
 - Checkpoint process (CKPT)
 - Log Writer process (LGWR)
 - System monitor process (SMON)
 - Listener registration process (LREG)
 - Process monitor process (PMON)
 - Recoverer process (RECO)
 - Archiver processes (ARCn) - optional
- **Server processes:** Oracle Database creates server processes to handle the connections to the database on behalf of client programs, and to perform the work for the client programs; for example, parsing and running SQL statements, and retrieving and returning results to the client program.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Multitenant Database



ORACLE®

A *database* is a set of physical files that store user data and metadata. The metadata consists of structural, configuration, and control information about the database server.

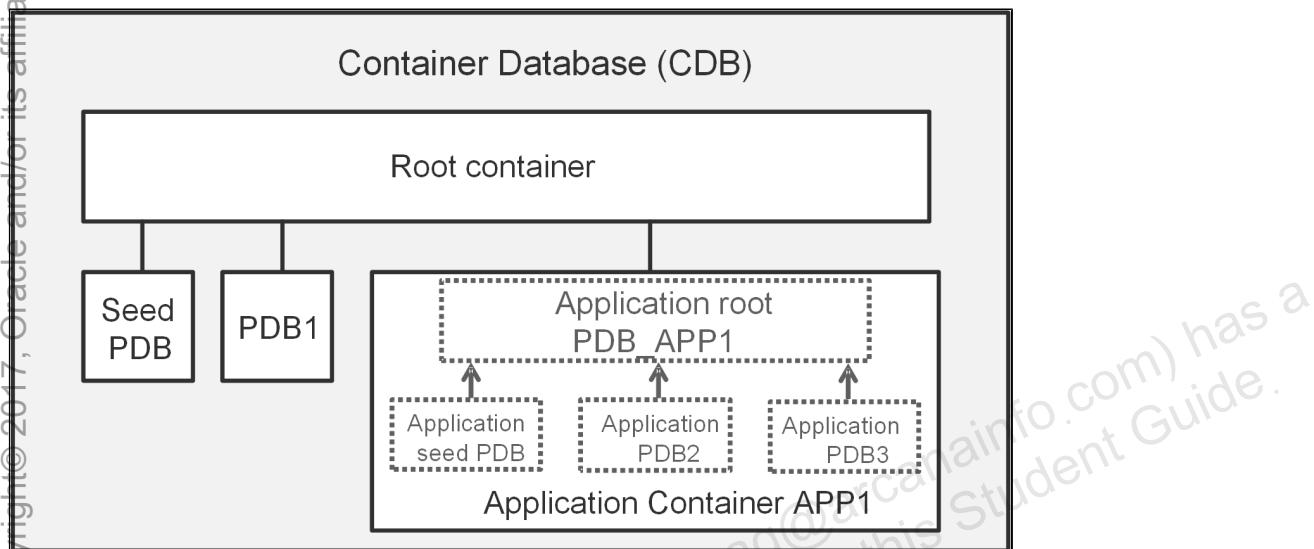
You can design your database to be a multitenant container database (CDB). A CDB, as illustrated above, is made up of one root container, one seed pluggable database (seed PDB), and one or more user-created pluggable databases (simply referred to as PDBs). To a user or application, PDBs appear logically as separate databases. You can still configure the database the old way using the non-container database (non-CDB) architecture, if needed.

- The root container, named CDB\$ROOT, contains multiple data files. The *data files* store Oracle-supplied metadata and common users (users that are known in every container). This information is shared with all PDBs.
- The seed PDB, named PDB\$SEED, is a system-supplied PDB template containing multiple data files that you can use to create new PDBs.
- The *user-created PDB* contains multiple data files that contain the data and code required to support an application; for example, a Human Resources application. Users interact only with the PDBs, and not the seed PDB or root container; for example, in the diagram above there are two PDBs - one for the sales organization (named SALES) and another for the Human Resources department (named HR). You can create multiple PDBs in a CDB. One of the goals of the multitenant architecture is that each PDB has a one-to-one relationship with an application.

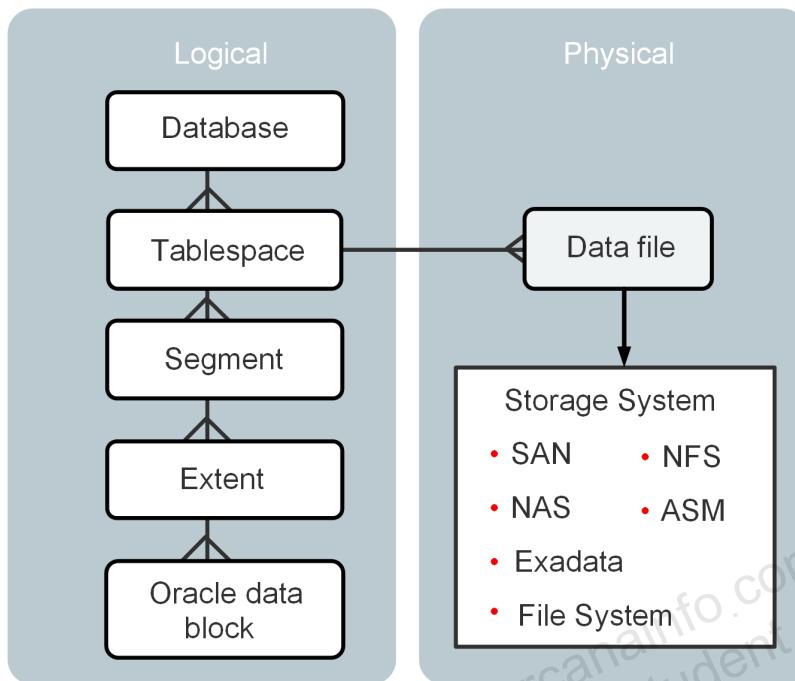
Application Containers

Oracle Database 12c Release 2 introduces the concept of an application container, which is a collection of PDBs within a CDB, storing data for an application. You create application containers to have a single master application definition. An application container, as shown below, consists of an application root container, an optional application seed PDB, and application PDBs.

Note: Application containers are covered briefly here so that you are aware of them. However, it is an advanced concept and is not practiced in this course.



Logical Storage Structures



ORACLE

Tablespaces, Segments, Blocks, and Extents

A database is divided into logical storage units called *tablespaces*, which collectively store all the database data. Each tablespace represents one or more data files, which can be stored in the following systems: storage area network (SAN), network attached storage (NAS), network file system (NFS), Exadata system, file system, or Oracle Automatic Storage Management. Each tablespace contains extents that store the data for logical objects, such as tables and indexes. These extents are collectively known as *segments*.

At the finest level of granularity, an Oracle database's data is stored in *data blocks*. One data block corresponds to a specific number of bytes of physical space on the disk. An *extent* is a specific number of contiguous Oracle data blocks (obtained in a single allocation) that are used to store a specific type of information. The diagram above illustrates the different levels of logical storage structures. You will learn more about these structures in [Lesson 6 Creating and Managing Tablespaces](#).

Default Tablespaces

The following table describes the tablespaces that get created for CDBs and PDBs by default. In short, the root container and PDBs each contain all the tablespaces listed below, with the exception that it is optional for a PDB to have an UNDO tablespace. A seed PDB has a SYSTEM, SYSAUX, and TEMP tablespace.

Tablespace	Description
SYSTEM	<p>The SYSTEM tablespace is used for core functionality. It stores the data dictionary (metadata that describes the objects in the database) and tables that contain administrative information about the database. All this information is contained in the SYS schema and can be accessed only by the SYS user or other administrative users with the required privilege. In the root container, it stores Oracle-supplied metadata, whereas in a PDB, it stores user metadata. Pointers from the PDBs to the Oracle-supplied objects allow the “system” objects to be accessed without duplicating them in the PDBs.</p>
SYSAUX	<p>The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace and helps to reduce the load on the SYSTEM tablespace. Some components and products that have used the SYSTEM tablespace or their own tablespaces in earlier releases of Oracle Database now use the SYSAUX tablespace. The SYSAUX tablespace exists in the root container and in each PDB.</p>
TEMP	<p>The TEMP tablespace contains schema objects only for a session's duration. Objects in temporary tablespaces are stored in temp files. Your temporary tablespace is used when you execute a SQL statement that requires the creation of temporary segments (such as a large sort or the creation of an index). Just as each user is assigned a default tablespace for storing created data objects, each user is assigned a temporary tablespace. By default, the root container has a single default temporary tablespace named TEMP that every PDB uses; however, if needed, you can create separate TEMP tablespaces in PDBs .</p>
UNDO	<p>The UNDO tablespace stores the data needed to roll back, or undo, changes to the database. In a single-instance CDB, one active UNDO tablespace exists in the root container. It is optional to have a local UNDO tablespace in a PDB.</p>
USERS	<p>The USERS tablespace stores user objects and data. If no default tablespace is specified when a user is created, then the USERS tablespace is the default tablespace for all objects created by that user. For the SYS and SYSTEM users, the default permanent tablespace is SYSTEM. The root container and each PDB has its own USERS tablespace.</p>

Database Files

- The following files do not belong to a container, but are used during the operation of the database:
 - Control file
 - Online redo log files
 - Initialization parameter file (pfile) or server parameter file (spfile)
 - Password file
 - Backup files
 - Archived redo log files
 - Alert log file
 - Trace files
- Data files, redo log files, and control files cannot be rebuilt unless they are recovered from database backups.

ORACLE®

The following files do not belong to a container like data files do, but are used during the operation of the database:

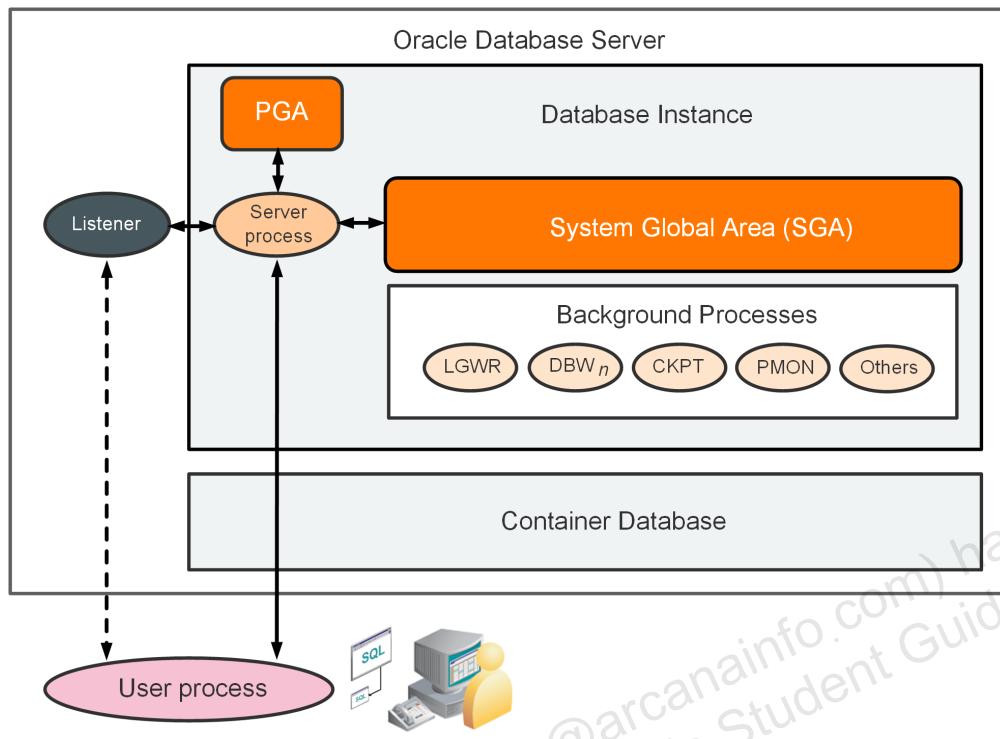
- **Control file:** This file stores metadata about the data files and online redo log files (for example, their names and statuses). This information is required by the database instance to open the database. Control files also contain metadata that must be accessible when the database is not open. It is highly recommended that you make several copies of the control file in your database server for high availability.
- **Online redo log files:** These files store changes to the database as they occur and are used for data recovery.
- **Initialization parameter file (pfile) or server parameter file (spfile):** This file defines how the database instance is configured when it starts up.
- **Password file:** This file enables users using the SYSDBA, SYSOPER, SYSBACKUP, SYSDG, SYSKM, SYSRAC, and SYSASM roles to connect remotely to the database instance and perform administrative tasks.
- **Backup files:** These files are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.
- **Archived redo log files:** These files contain an ongoing history of the data changes that are generated by the database instance. Using these files and a backup of the database, you can recover a lost data file. That is, archive logs enable the recovery of restored data files.
- **Alert log file:** These are special trace entries. The alert log of a database is a chronological log of messages and errors. Oracle recommends that you review the alert log periodically.
- **Trace files:** Each server process and background process can write to an associated trace file. When a process detects an internal error, the process dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, whereas other information is for Oracle Support Services.

Note: Data files reside in containers.

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Example - User Issuing a SQL Statement



ORACLE®

The following steps describe a basic Oracle database operation using memory, processes, and storage.

1. An instance has started on a node where Oracle Database is installed, often called the host or database server. The database is also opened after the instance has started.
2. A user starts an application spawning a user process. The application attempts to establish a connection to the server. The connection can be local, client/server, or a three-tier connection from a middle tier.
3. The server runs a listener that has the appropriate Oracle Net Services handler. The listener detects the connection request from the application and creates a dedicated server process on behalf of the user process. It is in *dedicated server mode* that a user process first communicates with a listener process.
4. The user runs a DML-type SQL statement and commits the transaction. For example, the user changes the address of a customer in a table and commits the change.
5. The server process receives the statement and checks the shared pool (an SGA component) for any shared SQL area that contains an identical SQL statement. If a shared SQL area is found, the server process checks the user's access privileges to the requested data, and the existing shared SQL area is used to process the statement. If a shared SQL area is not found, a new shared SQL area is allocated for the statement so that it can be parsed and processed.
6. The server process retrieves any necessary data values, either from the actual data file (table whose segment is stored on a data file) or from values already stored in the database buffer cache (an SGA component) because someone else had previously requested the same data.
7. The server process modifies data in the SGA. Because the transaction is committed, the Log Writer process (background process) immediately records the transaction in the online redo log file. The Database Writer process (another background process) writes modified blocks permanently to disk when it is efficient to do so.

8. If the transaction is successful, the server process sends a message across the network to the application. If it is not successful, an error message is transmitted.

Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

Connecting to Oracle Databases

- You connect client applications to an Oracle Database by connecting to its database instance, not its database.
 - The database instance accesses the CDB and PDBs for you.
- A user session is a logical entity that represents the state of the current user login to the database instance.
- Examples of connecting to Oracle Database:
 - By using operating system authentication
 - \$ sqlplus / as sysdba
 - By using Easy Connect Syntax
 - SQL> CONNECT hr/hr@host1.example.com:1521/db.example.com

ORACLE

Connections Versus Sessions

You connect client applications to database instances (not databases), which access the CDB and PDBs for you.

- A *connection* is the physical communication pathway between a client process and a database instance.
- A *user session* is a logical entity that represents the state of the current user login to the database instance. A session lasts from the time the user is authenticated by the database instance until the time the user disconnects or exits the client application.

Connecting to CDBs by Using Operating System Authentication

As a database administrator, you can quickly start SQL*Plus and connect to a root container without a password by using the following command. This command enables you to connect to the database as the `SYS` user with the `SYSDBA` privilege. There are some rules: You must be on the same machine as the database instance and the current operating system user must be a member of the privileged `OSDBA` group.

```
$ sqlplus / as sysdba
```

Connecting to PDBs by Using the Easy Connect Syntax

There are many ways to connect to a PDB, however, using the Easy Connect syntax in SQL*Plus is the easiest because it's already enabled on the database server by default and it doesn't require any client-side configuration. This syntax supports TCP protocol only (no SSL). It offers no support for advanced connection options such as connect-time failover, source routing, and load balancing.

Easy Connect connection strings take the following form:

```
SQL> CONNECT <username>/<password>@<listener hostname>:<listener port>/<service name>
```

For example, the SYSTEM user requests a connection to the database service named db.example.com. The listener is located on a machine named host01.example.com and listens on port 1521.

```
SQL> CONNECT hr/hr@host1.example.com:1521/db.example.com
```

If you're starting from a command prompt, then you can start sqlplus and log in at the same time:

```
$ sqlplus hr/hr@host1.example.com:1521/db.example.com
```

The listener port and service name are optional. If the listener port is not provided, Oracle Net assumes that the default port of 1521 is being used. If the service name is not provided, Oracle Net assumes that the database service name and host name provided in the connect string are identical. For example, assuming that the listener uses TCP to listen on port 1521, the connection string above can be shortened.

For example, a connection string like this:

```
SQL> CONNECT hr/hr@db.example.com:1521/db.example.com
```

...can be shortened like this:

```
SQL> CONNECT hr/hr@db.example.com
```

Disconnecting from the Database Instance

Use the EXIT command to exit SQL*Plus, disconnect from the database instance, and end all sessions in the database instance memory.

Oracle Database Tools

- Oracle Database tools each have their own purpose, and some operations you can perform in more than one tool.
- Finding the right tool for the job often comes down to preference, for example, whether you prefer to work with code or use a graphical user interface.
- Tools include:
 - SQL*Plus
 - SQL Command Line (SQLcl)
 - SQL Developer
 - Database Configuration Assistant (DBCA)
 - Oracle Enterprise Manager Database Express (EM Express)
 - Oracle Enterprise Manager Cloud Control
 - Others (netca, netmgr, adcri, ...)

ORACLE

The following table shows you at a glance which tool can perform which tasks.

	SQL*Plus	SQLcl	SQL Developer	DBCA	EM Database Express	EM Cloud Control	Oracle Universal Installer
Create a CDB or PDB	Yes	Yes	Yes (PDB only)	Yes	Yes (PDB only)	Yes (PDB only)	Yes
Explore CDB instance, architecture, and PDBs	Yes	Yes	Yes	No	Yes	Yes	No

SQLPlus

- Example 1: From a command line, you can start SQL*Plus, log in, and show the user that you're logged in as:

```
$ sqlplus / as sysdba
...
Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
SQL> show user
USER is "SYS"
```

- Example 2: Call a SQL script from the command line:

```
$ sqlplus hr/hr@HRPDB @script.sql
```

ORACLE

About SQL*Plus

SQL*Plus is a command-line program that you use to submit SQL and PL/SQL statements to an Oracle database. SQL*Plus is installed with Oracle Database and is located in the \$ORACLE_HOME/bin directory. You can start SQL*Plus from the command line, or from the Start menu on a Windows client. Use the SQL*Plus command-line interface to execute SQL*Plus, SQL, and PL/SQL commands to perform the following:

- Enter, edit, run, store, retrieve, and save SQL commands and PL/SQL blocks
- Format, calculate, store, and print query results
- List column definitions for any table
- Send messages to and accept responses from an end user
- Perform database administration

Calling a SQL Script from SQL*Plus

When calling a SQL script file from within SQL*Plus, you have the following options:

Option 1: Call the script from the command line when you first invoke SQL*Plus:

```
$ sqlplus hr/hr@HRPDB @script.sql
```

Option 2: Call the script from inside a SQL*Plus session simply by using the "@" operator:

```
SQL> @script.sql
```

When a script is saved from SQL*Plus by using the `SAVE` command, the `.sql` extension is automatically supplied. You can then execute the script without supplying the extension at execution time, for example:

```
SQL> @script
```

Calling SQL*Plus from a Shell Script

You can call SQL*Plus from a shell script or BAT file by invoking `sqlplus` and using the operating system scripting syntax for passing parameters. For example, suppose you have the following shell script:

```
# Name of this file: batch_sqlplus.sh
# Count employees and give raise.
sqlplus hr/hr <<EOF
select count(*) from employees;
update employees set salary = salary*1.10;
commit;
quit
EOF
```

You can call that shell script from the command line :

```
$ ./batch_sqlplus.sh
```

For More Information

To learn how to start SQL*Plus, see [Starting SQL Plus in SQL*Plus Users Guide and Reference](#).

SQLcl

- SQLcl
 - Is a tool that can access Oracle Databases
 - Is a cross between SQL*Plus and SQL Developer
 - Incorporates Linux-like and IDE features, and color
- Example query with formatting to fit the screen:

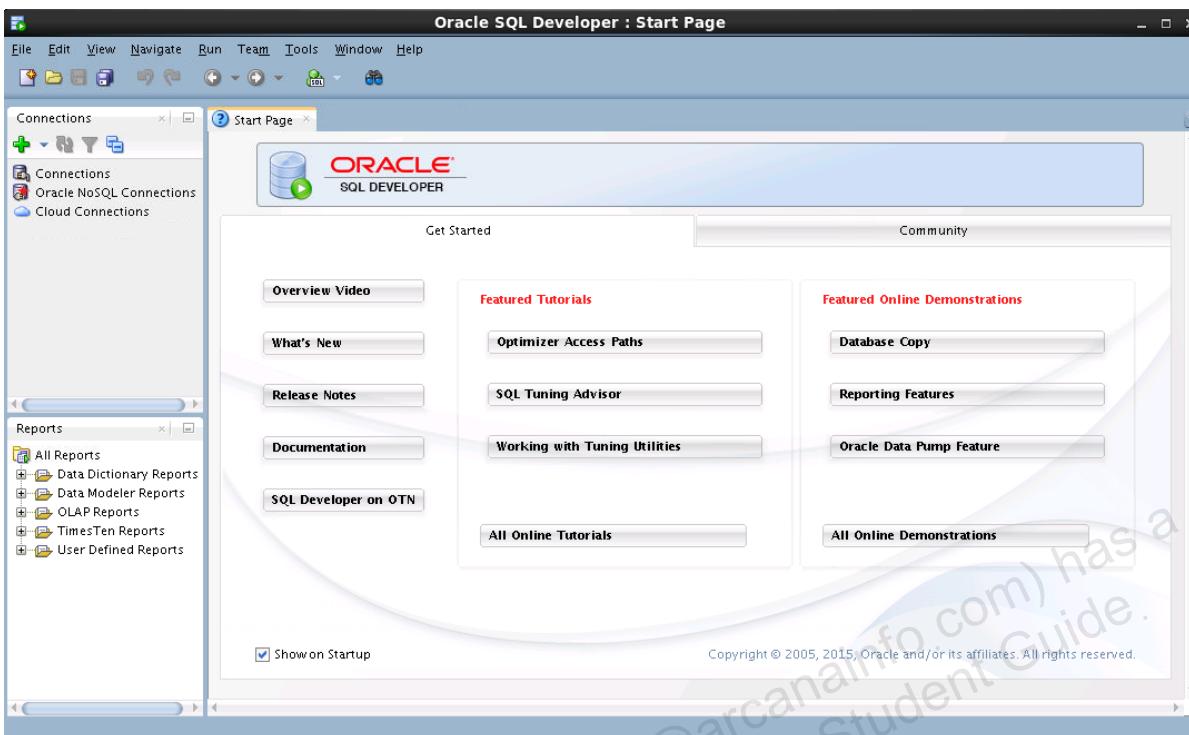
```
SQL> SELECT first_name, last_name, salary, hire_date FROM hr.employees;
```

FIRST_NAME	LAST_NAME	SALARY	HIRE_DATE
Steven	King	24000	17-JUN-03
Neena	Kochhar	17000	21-SEP-05
...			



Oracle SQL Developer

Unauthorized reproduction or distribution, in whole or in part, is illegal.



ORACLE

Oracle SQL Developer (SQL Developer) is a graphical tool for database developers and DBAs, and gets installed with Oracle Database. You can choose to display several different panes in the SQL Developer interface, such as a Connections and DBA pane (as shown in the image below). You use the former to define connections to databases and use the latter to perform DBA operations. You should add connections only for which the associated database user has DBA privileges or at least privileges for the desired DBA navigator operations on the specified database.

Some DBA-type operations that you can perform with SQL Developer include:

- Start and shut down PDBs
- Configure databases, for example, configure initialization parameters
- View the status of databases
- Export databases using Data Pump and import jobs
- Back up and recover data using Oracle Recovery Manager (RMAN)
- Configure Oracle Database Resource Manager (the Resource Manager), which enables you to manage multiple workloads within a database that are contending for system and database resources
- Schedule jobs, for example, loading data
- Configure security using audit settings, profiles, roles, and users
- Configure storage for archive logs, control files, data files, redo log groups, tablespaces, and temporary tablespace groups
- Run any number of provided reports, as well as create new ones

Unauthorized reproduction or distribution, in whole or in part, is illegal.

Some developer-type operations that you can perform with SQL Developer include:

- Develop scripts in both the SQL and PL/SQL languages
- Browse database objects
- Run SQL statements and SQL scripts
- Edit and debug PL/SQL statements

Database Configuration Assistant

Unauthorized reproduction or distribution prohibited. Copyright © 2017, Oracle and/or its affiliates.

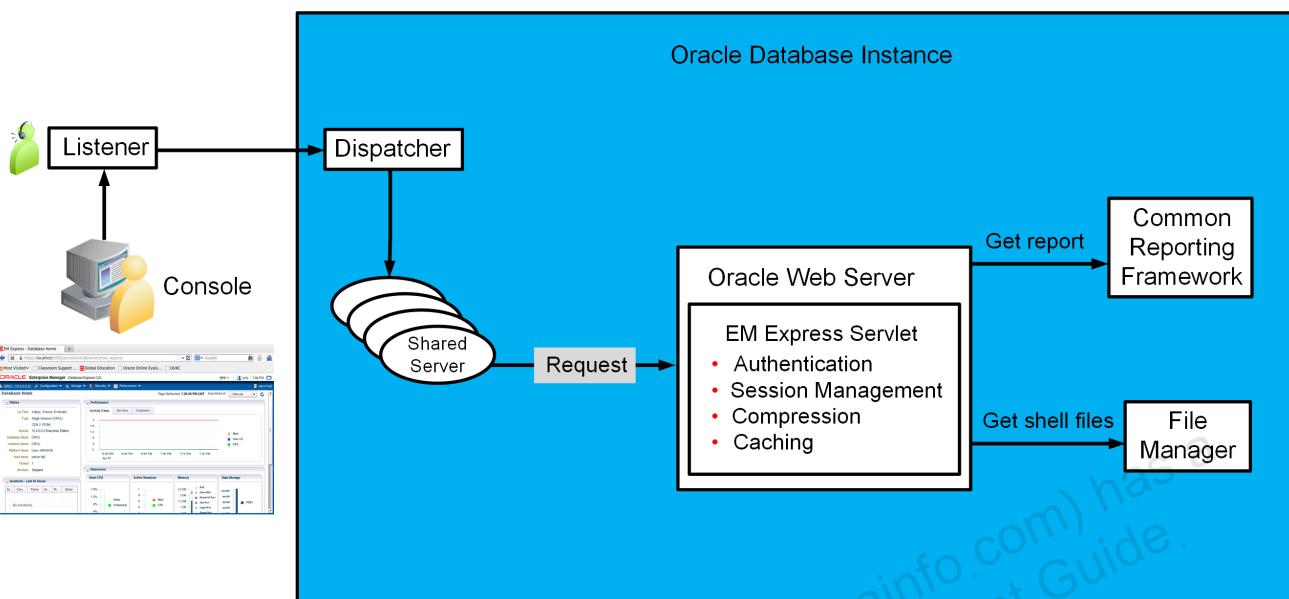


Database Configuration Assistant (DBCA) is a graphical tool that you can use to do the following:

- Create databases (CDBs and non-CDBs)
- Configure existing databases
- Delete databases
- Manage templates
- Manage pluggable databases
- Manage Oracle RAC database instances

Oracle Enterprise Manager Database Express

Unauthorized reproduction or distribution prohibited.



ORACLE®

About Oracle Enterprise Manager Database Express

Oracle Enterprise Manager Database Express (EM Express) is a lightweight tool that you can use to manage a CDB and all its PDBs (except the seed PDB). It provides an out-of-box browser-based management solution performance monitoring, configuration management, administration, diagnostics, and tuning.

Architecture

EM Express uses a web-based console, communicating with the built-in web server available in XML DB.

As requests from the console are processed, the EM Express servlet handles the requests, including authentication, session management, compression, and caching. The servlet passes requests for reports to the Common Reporting Framework and actions requiring shell files to the File Manager. This architecture is illustrated in the diagram above.

EM Express is available only when the database is open. This means that Enterprise Manager Database Express cannot be used to start up the database. Other operations that require that the database change state, such as enable or disable ARCHIVELOG mode, are also not available in EM Express.

EM Express is configurable with a single click in Database Configuration Assistant (DBCA).

EM Express is a servlet built on top of Oracle XML DB. The Oracle XML DB default wallet has a self-signed certificate, and some existing browsers consider self-signed certificates as untrusted because they are not signed by

a trusted CA (certificate authority). However, the self-signed certificate is still secure, as it ensures that the traffic is encrypted between the Oracle XML DB server and the client (browser). Therefore, enter a security exception for the EM Express URL in your web browser.

Requirements

The following are required for EM Express:

- XMLDB components must be installed on the Oracle Database server. All Oracle Databases of version 12.1.0 or higher have XMLDB installed.
- The web browser must have the Flash plug-in installed because EM Express uses Shockwave Flash (SWF) files.

Starting EM Express for CDBs and PDBs

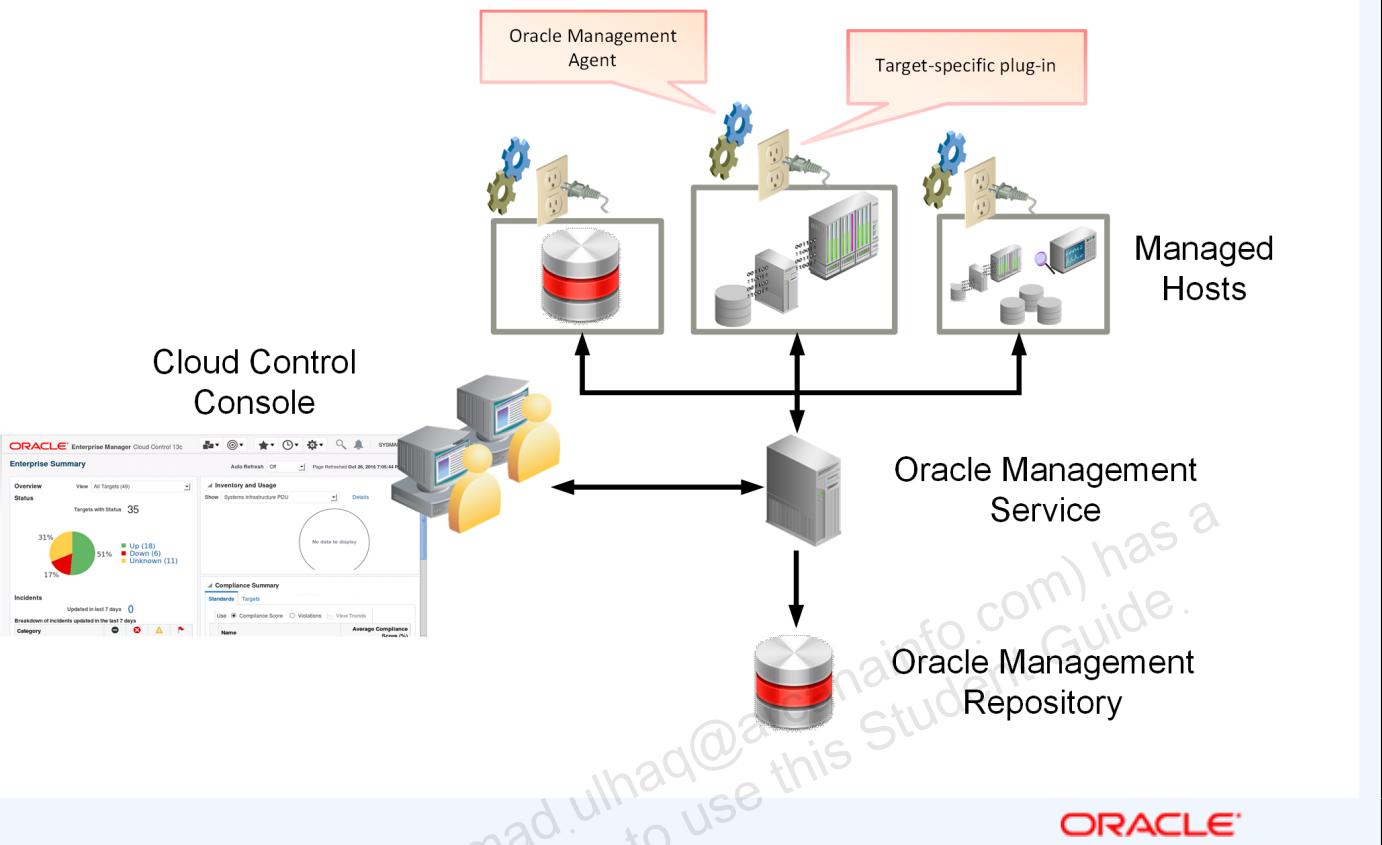
EM Express uses a global HTTPS port to connect to and manage non-CDBs, CDBs, and PDBs. Usually this port is provided by DBCA when it configures your non-CDB or CDB. By default, the HTTPS port that DBCA configures for a CDB can also be used for the PDBs in that CDB. You will, however, need to enable the global port before connecting to a PDB.

For more information, see:

- [Starting EM Express](#)
- [Configuring the HTTPS Port for EM Express](#)

Oracle Enterprise Manager Cloud Control

Unauthorized reproduction or distribution is illegal.



About Oracle Enterprise Manager Cloud Control

Oracle Enterprise Manager Cloud Control (EM Cloud Control) is Oracle's on-premises management platform, providing a single location for managing all your Oracle deployments, whether they be in your data centers or in the Oracle Cloud. Through deep integration with Oracle's product stack, EM Cloud Control provides management and automation support for Oracle applications, databases, middleware, hardware, and engineered systems.

Architecture

Oracle Enterprise Manager Cloud Control is composed of four main components, as illustrated in the diagram above:

- Oracle Management Repository (OMR)
- Oracle Management Service (OMS)
- Oracle Management Agent (OMA or agent) with target-specific plug-ins
- Cloud Control Console

The Oracle Management Agent runs on hosts, gathering metric data about those host environments as well as using plug-ins to monitor availability, configuration, and performance and to manage targets running on the host. The agents communicate with the Oracle Management Service to upload metric data collected by them and their plug-ins. In turn, the OMS stores the data it collects in the Oracle Management Repository where it can be accessed by the OMS for automated and manual reporting and monitoring. The OMS also communicates with the agents to

orchestrate the management of their monitored targets. As well as coordinating the agents, the OMS runs the Cloud Control Console web pages that are used by administrators and users to report on, monitor, and manage the computing environment that is visible to Cloud Control via the agents and their plug-ins.

Targets

Targets are the entities that EM Cloud Control manages. The following are supported targets:

- Oracle databases
- Oracle Database Listeners
- Oracle Fusion Middleware products
- Oracle Application Servers
- Oracle WebLogic Servers
- Oracle applications, including E-Business Suite, SOA, Siebel, and PeopleSoft
- Exadata and Exalogic
- Cloud Control Components: Oracle Management Repository (OMR) and Oracle Management Service (OMS)
- Third-party products

Cloud Control Console

The Enterprise Summary page in the Cloud Control Console includes the following:

- Information displayed in graphs and tables
- Summary information with drilldown capability to relevant details
- User-selected home page from a predefined set or based on any page in the console
- Menu-driven navigation
- Global target search
- History and favorites
- Customizable target home pages (per-user basis)

Oracle-Supplied User Accounts

- User types for Oracle Database include:
 - Database administrators
 - Security officers
 - Network administrators
 - Application administrators
 - Database users
- Oracle supplies the following user accounts:
 - Administrative user accounts (SYS, SYSTEM, SYSBACKUP, SYSDG, SYSKM, SYSRAC, SYSMAN, and DBSNMP)
 - Sample schema user accounts (HR)
 - Internal user accounts
- Users are common versus local:
 - CDB common user
 - Application common user
 - Local user

ORACLE®

Oracle Database User Types

The types of users and their roles and responsibilities depend on the database site. A small site can have one database administrator who administers the database for application developers and users. A very large site can find it necessary to divide the duties of a database administrator among several people and among several areas of specialization.

- **Database Administrators:** Each database requires at least one database administrator (DBA). An Oracle Database system can be large and can have many users. Therefore, database administration is sometimes not a one-person job, but a job for a group of DBAs who share responsibility.
- **Security Officers:** In some cases, a site assigns one or more security officers to a database. A security officer enrolls users, controls and monitors user access to the database, and maintains system security.
- **Network Administrators:** Some sites have one or more network administrators. A network administrator, for example, administers Oracle networking products, such as Oracle Net Services.
- **Application Administrators:** An Oracle Database site can assign one or more application administrators to administer a particular application. Each application can have its own administrator.
- **Database Users:** Database users interact with the database through applications or utilities.

Oracle-Supplied User Accounts

Oracle supplies several predefined user accounts in an Oracle Database:

- Administrative user accounts are created so that you can administer the database: SYS, SYSTEM, SYSBACKUP,

SYSDG, SYSKM, SYSRAC, SYSMAN, and DBSNMP. The SYSMAN account gets created if you register the database to Cloud Control during installation, and is used to perform Cloud Control administration tasks. The management agent of Cloud Control uses the DBSNMP account to monitor and manage the database. You must not delete these accounts.

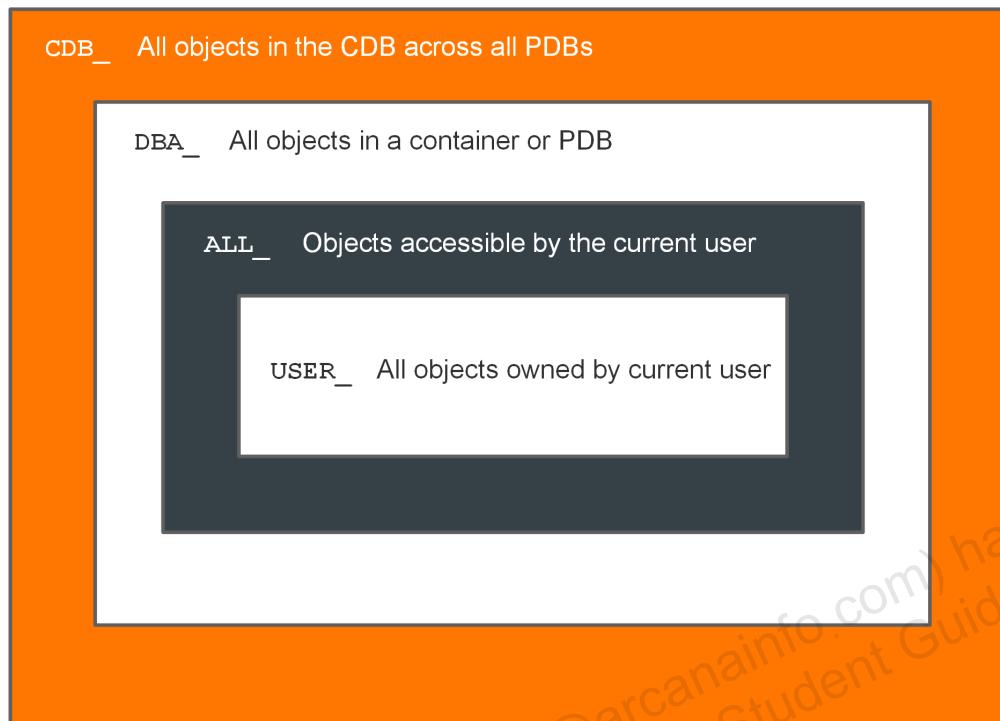
- Sample schema user accounts are used for examples in Oracle Database documentation and instructional materials, for example, the HR account. You must unlock this type of account and reset its password before using it.
- Internal user accounts are created specifically for Oracle Database features and components. You must not delete internal accounts, and you must not attempt to log in with them.

Common Versus Local Users

In a CDB, a user account is either common or local:

- A *CDB common user* is a database account that is created in the root container and is inherited by all PDBs in the CDB, including future PDBs and all application containers. A common user cannot have the same name as any local user across all of the PDBs. All Oracle-supplied administrative user accounts, such as SYS and SYSTEM , are common users. Other than Oracle-supplied administrative accounts, all common user account names start with a user-defined prefix, such as C##; for example, C##JODY.
- An *application common user* is a database account that is created in an application root and is inherited by all the application PDBs associated to the application root (including future application PDBs). An application common user is not inherited in the root container.
- A *local user* is a database user that is created in a specific PDB. You can create the same account and password in more than one PDB.

Querying the Oracle Data Dictionary



ORACLE®

The *Oracle data dictionary* is the metadata of the database and contains the names and attributes of all objects in the database. The creation or modification of any object causes an update to the data dictionary that reflects those changes. This information is stored in the base tables that are maintained by the Oracle Database server, but you access these tables by using predefined views rather than reading the tables directly.

The data dictionary:

- Is used by the Oracle Database server to find information about users, objects, constraints, and storage
- Is maintained by the Oracle Database server as object structures or definitions are modified
- Is available for use by users to query information about the database
- Is owned by the `SYS` user
- Should never be modified directly using SQL

CDB_, DBA_, ALL_, and USER_ Views

The view prefixes, as shown in the diagram above, indicate the data (and how much of that data) a given user can see.

- `CDB_` views contain metadata for all objects in a CDB across all PDBs.
- `DBA_` views contain metadata for all objects in a container or PDB.
- `ALL_` views contain metadata for objects that the current user is privileged to see, whether the user owns them or not. For example, if `USER_A` has been granted access to a table owned by `USER_B`, then `USER_A` sees that table listed in any `ALL_` view dealing with table names.

- `USER_` views contain metadata for all objects owned by the current user; that is, objects that are present in the user's own schema.

Only `USER_` and `ALL_` views are available to any user. The `CDB_` and `DBA_` views are restricted to DBA accounts.

Generally, each view set is a subset of the higher-privileged view set, row-wise and column-wise. Not all views in a given view set have a corresponding view in the other view sets. It depends on the nature of the information in the view. For example, there is a `DBA_LOCK` view, but no `ALL_LOCK` view, because only a DBA would have interest in data about locks. Be sure to choose the appropriate view set to meet the need that you have. If you have the privilege to access the DBA views, you still may want to query only the `USER_` version of the view because the results show information on objects that you own and you may not want other objects to be added to your result set.

The `CDB_` and `DBA_` views can be queried only by users with the `SYSDBA` or `SELECT ANY DICTIONARY` privilege, or `SELECT_CATALOG_ROLE` role, or by users with direct privileges granted to them. Many applications require grants to one or more individual dictionary views.

Summary for Lesson 1

In this lesson, you should have learned how to:

- Describe Oracle Database and its editions
- Explain the general structure of a relational database model
- Describe SQL and PL/SQL
- Describe the architecture of a multitenant Oracle Database
- Connect to an Oracle Database by using OS authentication and Easy Connect Syntax
- Describe the main Oracle Database tools
- Describe the Oracle-supplied user accounts for Oracle Database
- Query the data dictionary in an Oracle Database



ORACLE

Samad Ul Haq (samad.ulhaq@arcanainfo.com)
non-transferable license to use this material

Practice 1 Overview

- 1-1: Exploring a CDB with SQL*Plus
- 1-2: Exploring a PDB with SQL*Plus
- 1-3: Exploring a CDB and PDB with EM Express
- 1-4: Getting Started with SQLcl

Practice 1-1 Exploring a CDB

Overview

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

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



Tip:

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

- Log in to the root container as a common user. A *CDB common user* is a database account that is created in the root container and is inherited by all PDBs in the CDB.
- Query container data objects, such as views whose names begin with `v$` and `CDB_`.

For more information, refer to the following sections in *Oracle Database Administrator's Guide*:

- [About Viewing Information When the Current Container is the CDB Root](#)
- [Viewing Information About the Containers in a CDB](#)

In some of the steps below, you will format columns by using the `COLUMN` command. For example, applying the format `A55` specifies an alphabetic format of 55 characters wide. Format `999` is an example of a numeric format. See `COLUMN` in *SQL*Plus User's Guide and Reference*.

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

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

Assumptions

You are logged in to VM1 as the `oracle` user. VM1 is one of your two virtual machines that you will use throughout this course to complete the practices. Its host name is `12cr2db`.

Tasks

Complete the following steps on VM1.

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

```
$ which oraenv
/usr/local/bin/oraenv
$
```

- b. Source the `oraenv` script. The dot in the command means to do a *source* operation. `Oraenv` sets the required environment variables needed for you to connect to your database instance. For example, it sets the `ORACLE_SID` and `ORACLE_HOME` environment variables and includes the `$ORACLE_HOME/bin` directory in the `PATH` environment variable setting. Environment variables that this script sets will persist in the terminal window until you close it. Don't forget to put a space after the dot. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

- c. View the environment variables set by the `oraenv` command that you just ran.

```
$ set | grep ORACLE
OLD_ORACLE_BASE=
ORACLE_BASE=/u01/app/oracle
ORACLE_HOME=/u01/app/oracle/product/12.2.0/dbhome_1
ORACLE_SID=ORCL
$
```

Therefore, remember, that each time from this point on, when you open a terminal window, you will need to source the `oraenv` script to set the environment variables for your CDB.

2. Connect to the root container by using SQL*Plus.

- a. Start SQL*Plus and log in to the root container of your CDB as the `SYS` user with the `SYSDBA` privilege. In the command below, you must use lowercase letters for `sqlplus`. You can connect to a database without a password when you have a local connection (on the same machine) and the current operating system user is a member of the privileged `OSDBA` group. On VM1, the `OSDBA` group is named `dba`, and the user `oracle` is a member of that group. Because you are currently connected to the operating system as the `oracle` user, you can connect to the database as the `SYS` user without a password. Including `as sysdba` in the command enables you to perform all administrative tasks in the database. For example, you can create, drop, open, mount, start up, and shut down a database.

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 14 19:42:08 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
SQL>
```

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

The `SHOW` command is a tool-specific command, and is supported in SQL*Plus, SQL*Developer, and SQLcl. It is not SQL language, does not appear in any SQL Language reference manuals, and is not available to any tool or utility that relies on the SQL Language like [Toad](#) or within PL/SQL programs or Java programs. For this example and others that follow, you could use SQL instead of the `SHOW` command, if desired.

```
SQL> SHOW user
```

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

3. View information about the containers in your CDB.

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

```
SQL> SHOW con_name
```

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

- Show the current container id.

Because you're currently connected to the root container, the id should be 1.

```
SQL> SHOW con_id
```

```
CON_ID
-----
1
SQL>
```

- Verify that you do indeed have a container database, as opposed to a non-container database by querying the `V$DATABASE` view. The name `ORCL` should be listed in the `NAME` column, `YES` should be listed in the `CDB` column, and the `ID` should be 0 (zero). A value of zero is used for rows containing data that pertain to the entire CDB. This value is also used for rows in non-CDBs.

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

NAME	CDB	CON_ID
ORCL	YES	0

- Show the version of your Oracle Database by querying the `V$VERSION` view. This view displays version numbers of core library components in the Oracle Database. The results should indicate that your database is version 12.2.0.1.0.

Tip: If you're ever confused about whether you're connected to VM1 or VM2 in the classroom, VM1 has a 12.2 database installed, whereas VM2 has a 12.1 database.

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

BANNER
-----
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0
- 64bit Production
PL/SQL Release 12.2.0.1.0 - Production
CORE    12.2.0.1.0      Production
TNS for Linux: Version 12.2.0.1.0 - Production
NLSRTL Version 12.2.0.1.0 - Production
SQL>
```

- e. List all the containers in your CDB by querying the [V\\$CONTAINERS](#) view. The results should list three containers - the root container (CDB\$ROOT), the seed PDB (PDB\$SEED), and PDB1.

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

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

- f. List the PDBs in the CDB by using the SHOW command. The query should list two PDBs - the seed PDB (PDB\$SEED) and PDB1.

Note: You can also list PDBs by querying the V\$PDBS view.

The SHOW command includes information about the open mode of each PDB and whether the PDB is restricted. The *open mode* for a PDB determines what type of activities a PDB will allow at that time, for example, queries and data changes. See [About the Open Mode of a PDB](#) in *Oracle Database Administrator's Guide* for definitions of each open mode. PDB\$SEED is in READ ONLY mode and PDB1 is in READ WRITE mode.

The RESTRICTED column indicates whether only users possessing the RESTRICTED SESSION privilege can connect to the PDB.

```
SQL> SHOW pdbs;

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

- g. View the status of all PDBs in the CDB by querying the CDB_PDBS view.

The *status* of a PDB describes the state of the PDB; for example, if the PDB is new, but never opened (NEW), or if it is available and ready for use (NORMAL). See [DBA_PDBS](#) in the *Oracle Database Reference Guide* for definitions of each status.

```

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

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

```

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

- List all the data files in the CDB (for the root container and all PDBs) by querying the CDB_DATA_FILES view. The order of your results may vary.

A *data file* is a physical storage structure, and each data file is associated with only one tablespace. Multiple data files might constitute a single tablespace.

```

SQL> COLUMN file_name FORMAT A50
SQL> COLUMN tablespace_name FORMAT A10
SQL> SELECT file_name, tablespace_name FROM cdb_data_files;

FILE_NAME                                TABLESPACE
-----  -----
/u01/app/oracle/oradata/ORCL/users01.dbf    USERS
/u01/app/oracle/oradata/ORCL/undotbs01.dbf   UNDOTBS1
/u01/app/oracle/oradata/ORCL/system01.dbf    SYSTEM
/u01/app/oracle/oradata/ORCL/sysaux01.dbf    SYSAUX
/u01/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf  UNDOTBS1
/u01/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf  SYSAUX
/u01/app/oracle/oradata/ORCL/PDB1/system01.dbf  SYSTEM
/u01/app/oracle/oradata/ORCL/PDB1/users01.dbf   USERS
SQL>

```

- List all the tablespaces in the CDB (for both the root container and all the PDBs) by querying the V\$DATAFILE and V\$TABLESPACE views. In the SQL statement below, the FROM clause includes short aliases for the table names. For example, the alias for V\$DATAFILE is d. When you query multiple tables, you must identify the table for each column by prefixing the columns with either the full table name or the short alias name. In this example, notice that aliases are used in the SELECT and WHERE clauses.

A *tablespace* is a logical storage structure that consists of one or more data files. The V\$TABLESPACE and V\$DATAFILE views both contain tablespace information from the control file.

```

SQL> COL name FORMAT A12
SQL> SELECT d.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
   ORDER BY 4;

  FILE# NAME          TS#    CON_ID
  -----
    3  SYSAUX         1        1
    1  SYSTEM          0        1
    4  UNDOTBS1        2        1
    7  USERS           4        1
    5  SYSTEM          0        2
    6  SYSAUX          1        2
    8  UNDOTBS1        2        2
    9  SYSTEM          0        3
   10  SYSAUX          1        3
   11  UNDOTBS1        2        3
   12  USERS           4        3
11 rows selected.
SQL>

```

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

A temp file is a special data file that is associated with a temporary tablespace. Oracle Database uses temporary tablespaces to store global temporary tables and to perform tasks that are best not done in memory like sorting a large result set or performing a complex query join.

```

SQL> SELECT file_name, tablespace_name FROM cdb_temp_files;

FILE_NAME                      TABLESPACE
-----
/u01/app/oracle/oradata/ORCL/temp01.dbf      TEMP
/u01/app/oracle/oradata/ORCL/PDB1/temp01.dbf  TEMP
SQL>

```

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

The redo log for a database consists of two or more *redo log files*. The database requires a minimum of two files to guarantee that one is always available for writing while the other is being archived (if the database is in `ARCHIVELOG` mode). The Log Writer process (LGWR) writes to redo log files in a circular fashion. When the current redo log file fills, LGWR begins writing to the next available redo log file. When the last available redo log file is filled, LGWR returns to the first redo log file and writes to it, starting the cycle again.

The `CON_ID` values in the query below are equal to zero below because the redo log files exist for the whole instance and not for particular containers.

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

GROUP# MEMBER	CON_ID
3 /u01/app/oracle/oradata/ORCL redo03.log	0
2 /u01/app/oracle/oradata/ORCL redo02.log	0
1 /u01/app/oracle/oradata/ORCL redo01.log	0

SQL>

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

A *database control file* is a small binary file necessary for the database to start and operate successfully. A control file is associated with only one Oracle Database. DBCA creates two control files when it creates a CDB. Both files are very active while the database is running. The `CON_ID` values in the query below are equal to zero because the control files exist for the whole instance and not for particular containers.

```
SQL> COLUMN name FORMAT A55
SQL> SELECT name, con_id FROM v$controlfile;
NAME                                     CON_ID
-----                                     -----
/u01/app/oracle/oradata/ORCL/control01.ctl      0
/u01/app/oracle/fast_recovery_area/ORCL/control02.ctl 0
SQL>
```

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

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

A *common user* is a database user that has the same identity in the root container and in every existing and future PDB. Every common user can connect to and perform operations within the root container, and within any PDB in which it has privileges. Every common user is either Oracle-supplied or user-created. Examples of Oracle-supplied common users are `SYS` and `SYSTEM`.

```

SQL> SET PAGES 140
SQL> SELECT DISTINCT username FROM cdb_users WHERE common = 'YES' ORDER BY 1;

USERNAME
-----
ANONYMOUS
APPQOSSYS
AUDSYS
CTXSYS
DBSFWUSER
DBSNMP
DIP
DVF
DVSYS
FLOWS_FILES
GGSYS
GSMADMIN_INTERNAL
GSMCATUSER
GSMUSER
LBACSYS
MDDATA
MDSYS
OJVMSYS
OLAPSYS
ORACLE_OCM
ORDDATA
ORDPLUGINS
ORDSYS
OUTLN
REMOTE_SCHEDULER_AGENT
SI_INFORMTN_SCHEMA
SPATIAL_CSW_ADMIN_USR
SYS
SYS$UMF
SYSBACKUP
SYSDG
SYSKM
SYSRAC
SYSTEM
WMSYS
XDB
XS$NULL
36 rows selected.
SQL>

```

- b. List all the users in every PDB in the CDB by querying the CDB_USERS view. In the results, notice that the SYS, SYSTEM, HR (for the sample data), and PDBADMIN user accounts are listed for PDB1. The root container's id is 1 and PDB1's id is 3. PDB\$SEED is an Oracle-managed template database for default provisioning. For this reason, it is ignored/filtered in the management views. Note: The order of your results may differ than the results below.

```
SQL> COLUMN username FORMAT A25
```

```
SQL> SELECT con_id, username FROM cdb_users;

CON_ID USERNAME
-----
1   SYS
1   SYSTEM
1   XS$NULL
1   OJVMSYS
1   LBACSYS
1   OUTLN
1   SYS$UMF
1   DBSNMP
1   APPQOSSYS
1   GGSYS
1   ANONYMOUS
1   CTXSYS
1   SI_INFORMTN_SCHEMA
1   DVSYS
1   DVF
1   GSMADMIN_INTERNAL
1   ORDPLUGINS
1   MDSYS
1   OLAPSYS
1   ORDDATA
1   XDB
1   WMSYS
1   ORDSYS
1   GSMCATUSER
1   MDDATA
1   SYSBACKUP
1   REMOTE_SCHEDULER_AGENT
1   DBSFWUSER
1   GSMUSER
1   SYSRAC
1   AUDSYS
1   DIP
1   SYSKM
1   ORACLE_OCM
1   SYSDG
1   SPATIAL_CSW_ADMIN_USR
3   SYS
3   SYSTEM
3   XS$NULL
3   LBACSYS
3   OUTLN
3   DBSNMP
3   APPQOSSYS
3   GGSYS
3   ANONYMOUS
3   HR
3   CTXSYS
3   SI_INFORMTN_SCHEMA
3   DVSYS
3   DVF
3   GSMADMIN_INTERNAL
3   ORDPLUGINS
```

```

3 MDSYS
3 OLAPSYS
3 ORDDATA
3 XDB
3 WMSYS
3 ORDSYS
3 GSMCATUSER
3 MDDATA
3 SYSBACKUP
3 REMOTE_SCHEDULER_AGENT
3 DBSFWUSER
3 PDBADMIN
3 GSMUSER
3 SYSRAC
3 OJVMSYS
3 AUDSYS
3 DIP
3 SYSKM
3 ORACLE_OCM
3 SYS$UMF
3 SYSDG
3 SPATIAL_CSW_ADMIN_USR
SQL>

```

6. View information about the database instance and the services.

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

INSTANCE_NAME	STATUS	CON_ID
ORCL	OPEN	0

- b. List the services for all the containers in the CDB by querying the `V$SERVICES` view. The query returns the following five services. The `PDB$SEED` service is not listed because no one should connect to it and no operation should be performed with it. It is reserved as a template to create other PDBs.
- a. `ORCLXDB`: This is an internal service for Oracle XMLDB (XDB). You use Oracle XML DB to store, generate, manipulate, manage, and query XML data in the database. Oracle Enterprise Manager Express 12c is developed using Oracle XML DB.
 - b. `ORCL.example.com`: This is the service for the CDB.
 - c. `SYS$USERS`: This internal service is for user sessions that are not associated with services.
 - d. `SYS$BACKGROUND`: This internal service is used by Oracle background processes only.
 - e. `pdb1.example.com`: This is the service for the PDB.

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

  CON_ID NAME
  -----
    1 ORCLXDB
    1 ORCL.example.com
    1 SYS$USERS
    1 SYS$BACKGROUND
    3 pdb1.example.com

SQL>
```

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

```
SQL > EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 1-2 Exploring a PDB

Objectives

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

- Connect to a PDB indirectly through a CDB
- Query the data dictionary to view information about data files, temp files, and users in a PDB
- Connect to a PDB directly by using the Easy Connect syntax

Note: Some of the results in the Tasks section have been formatted for easier viewing.

Tip:

To find data dictionary information specific to a root container or a PDB:

- Query DBA_ views to return container-specific information.
- When you're logged into a PDB, queries against the data dictionary return information about that PDB only, regardless of the view you query. For example, you'll get the same results if you query DBA_DATA_FILES or its corresponding CDB_DATA_FILES view, even though CDB_DATA_FILES contains more data than DBA_DATA_FILES.
- When queried from a PDB, the DBA_PDBS view returns the information related to the PDB to which you are connected. When queried from the root container, the DBA_PDBS view describes all PDBs belonging to a given CDB.

Assumptions

You are logged in to VM1 as the oracle user.

Tasks

Complete the following steps on VM1.

1. Connect to PDB1 indirectly through the root container.

- a. Open a new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

- b. Start SQL*Plus and connect to the root container as the SYS user with the sysdba privilege. Oracle allows any DBA group user at the operating system level to log into SQL*Plus without any authentication.

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Fri Jul 15 19:17:37 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
SQL>
```

- c. Verify that PDB1 is open. After DBCA creates a PDB, it opens it automatically. The results below indicate that PDB1's open mode is READ WRITE, which means PDB1 is open.

PDB users with the SYSDBA, SYSOPER, SYSPBACKUP, SYSDG, SYSKM, or SYSRAC privilege can connect to a closed PDB; however, all other PDB users can connect only when the PDB is open.

Note: You'll learn more about open modes in [Opening and Closing PDBs](#) in Lesson 2.

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

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

SQL>
```

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

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

If PDB1 is already open, you will get the following message:

```
*  
ERROR at line 1:  
ORA-65019: pluggable database PDB1 already open
```

- e. Switch to PDB1.

When logged in to a CDB as an appropriately privileged user, you can use the ALTER SESSION command to switch between containers within the CDB.

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

From this point on, your queries against the data dictionary will retrieve information for PDB1 only.

- f. Verify that the container name is PDB1.

```
SQL> SHOW con_name
```

CON_NAME

PDB1

SQL>

2. Query the data dictionary to list the data files and temp files for PDB1. The results are formatted for easier viewing.

Note: The steps below could be performed without switching containers by querying the CDB_DATA_FILES view while displaying the CON_ID column.

- a. List the data files for PDB1 and the tablespaces to which they belong by querying the DBA_DATA_FILES view.

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

FILE_NAME

TABLESPACE_NAME

/u01/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf	UNDOTBS1
/u01/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf	SYSAUX
/u01/app/oracle/oradata/ORCL/PDB1/system01.dbf	SYSTEM
/u01/app/oracle/oradata/ORCL/PDB1/users01.dbf	USERS

SQL>

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

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

FILE_NAME

TABLESPACE_NAME

/u01/app/oracle/oradata/ORCL/PDB1/temp01.dbf	TEMP
--	------

SQL>

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

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

USERNAME

PDBADMIN

HR

SQL>

3. Make a direct connection to PDB1 by using the Easy Connect syntax.

The Easy Connect syntax enables you to connect to the PDB without 1) requiring a connection to the root container, and 2) having to set up a net service name for the PDB. You'll learn how to set up net service names later in the course.

- a. Disconnect from the PDB.

```
SQL > DISCONNECT
```

```
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0
- 64bit Production
SQL>
```

You are still in SQL*Plus, but no longer connected to the database as a user.

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

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

- c. Connect to PDB1 directly by using the Easy Connect syntax. See [Appendix - Product-Specific Credentials](#) for the SYSTEM user password.

```
SQL> CONNECT system/<password>@localhost:1521/PDB1.example.com
Connected.
SQL>
```

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

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

- e. Exit SQL*Plus and close the terminal window.

```
SQL> EXIT
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0
- 64bit Production
$
```

Practice 1-3 Exploring a CDB and PDB with EM Express

Objectives

In this practice, you practice the following things:

- Determine the HTTPS port number on which your CDB listens to Oracle Enterprise Manager Database Express (EM Express)
- Enable the global port for your CDB so that you can start EM Express for a PDB
- Start EM Express for your CDB
- Start EM Express for PDB1
- Explore the differences in the EM Express interface when connected to a CDB versus a PDB

Tip:

Although you can connect to EM Express by using the `SYS` account, Oracle recommends that you create a named user account specifically for administering the Oracle database. This way, you can monitor database activity. For now, however, you'll log in with the `SYS` account. To back up, recover, or upgrade the database, you must log in as a user with the `SYSDBA` or `SYSBACKUP` privilege.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Complete the following steps on VM1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Determine the port number on which your CDB listens to EM Express.

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

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Fri Jul 15 20:11:37 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
SQL>
```

- b. Execute the `DBMS_XDB_CONFIG.GETHTTPSPORT` procedure to view the HTTPS port number

automatically configured on your CDB by DBCA for EM Express. The query should return port 5500. A value of zero indicates that an EM Express port is not configured and you'll need to manually configure one. All communication between your CDB and EM Express is encrypted, by default. The DUAL table has one column, DUMMY, defined to be VARCHAR2(1), and contains one row with a value x. Selecting from the DUAL table is useful for computing a constant expression with the SELECT statement.

Note: You can execute the DBMS_XDB_CONFIG.GETHTTPSPORT procedure to check for unencrypted protocols being used. This is good for security audits and will return a port number of zero.

```
SQL> SELECT dbms_xdb_config.gethttpsport() FROM dual;  
DBMS_XDB_CONFIG.GETHTTPSPORT()  
-----  
      5500  
SQL>
```

3. Enable the global port for the CDB.

```
SQL> exec dbms_xdb_config.setGlobalPortEnabled(TRUE);  
  
PL/SQL procedure successfully completed.  
SQL>
```

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

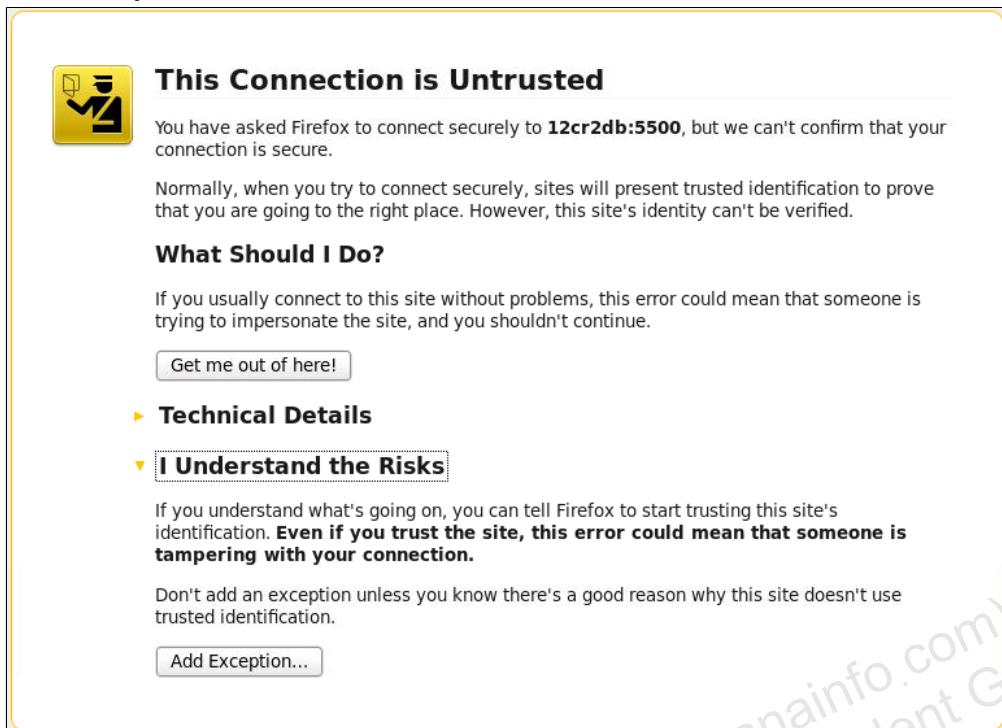
```
SQL> EXIT  
  
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -  
64-bit Production  
$
```

5. Start EM Express for a CDB. To do this, open a Firefox web browser, and enter the URL <https://localhost:5500/em>.

The first time you enter the URL for EM Express in your web browser, your browser may display warning messages. EM Express is a servlet built on top of Oracle XML DB. The Oracle XML DB default wallet has a self-signed certificate, and some existing browsers consider self-signed certificates as untrusted because they are not signed by a trusted certificate authority. However, the self-signed certificate is still secure, as it ensures that the traffic is encrypted between the Oracle XML DB server and the client (browser). Therefore, you'll need to enter a security exception for the EM Express URL in your web browser.

6. If needed, enter a security exception for the EM Express URL. VM1 is set up so that you don't have to enter a security exception.

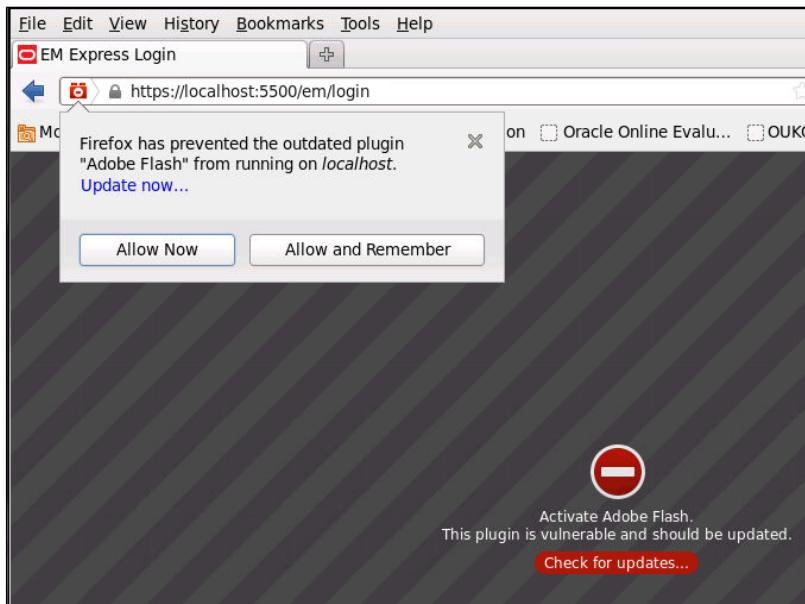
- a. On the "This Connection is Untrusted" page is displayed, expand **I Understand the Risks**, and click **Add Exception**.



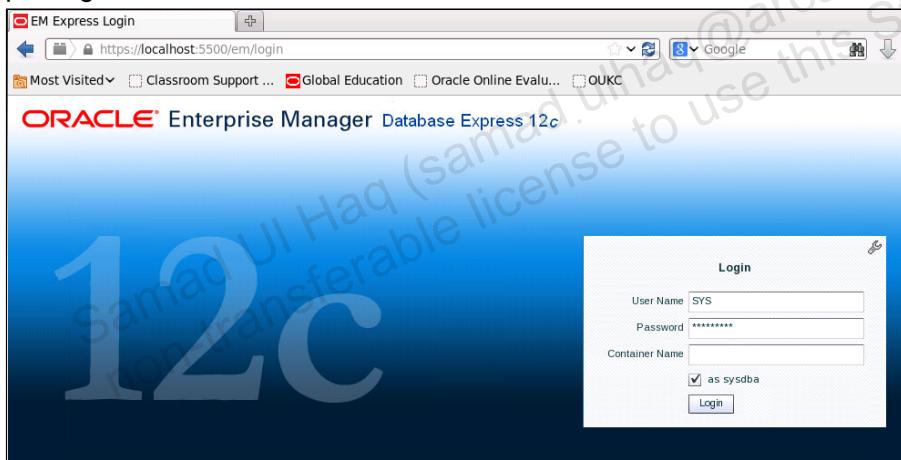
- b. In the Add Security Exception dialog box, retain the default settings, and click **Confirm Security Exception**.



- c. To the left of the URL, click the icon, and then click **Allow and Remember**.

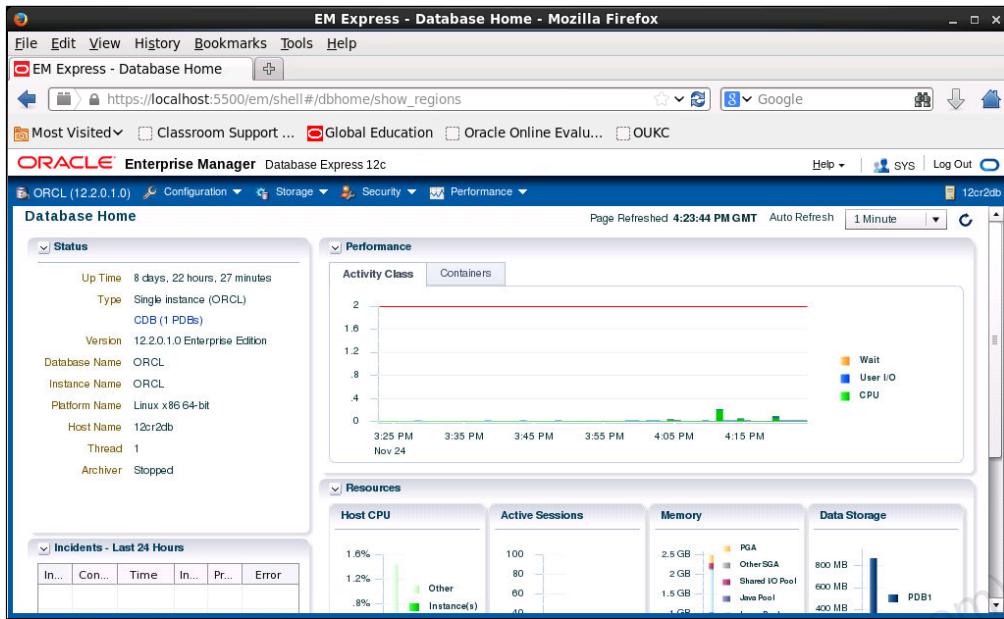


7. On the Login page for EM Express, enter the user name **sys** and the password as specified in [Appendix - Product-Specific Credentials](#). Leave the Container Name box empty. Select the **as sysdba** check box. Click **Login**. Remember, to manage a root container with EM Express, you must log in as a user with the SYS DBA privilege.



8. View the EM Express Home page for the CDB. The Status column provides information about:

- Up time duration
 - Instance type (single instance named ORCL; a CDB with 1 PDB)
 - Database version (12.2.0.1.0)
 - Database name (ORCL)
 - Instance name (ORCL)
 - Platform name (Linux x86 64-bit)
 - Host name (12cr2db)
 - Thread number (1)
 - Whether the archiver process is running or stopped (currently stopped)
- Also notice that you can view configuration, storage, security, and performance information.

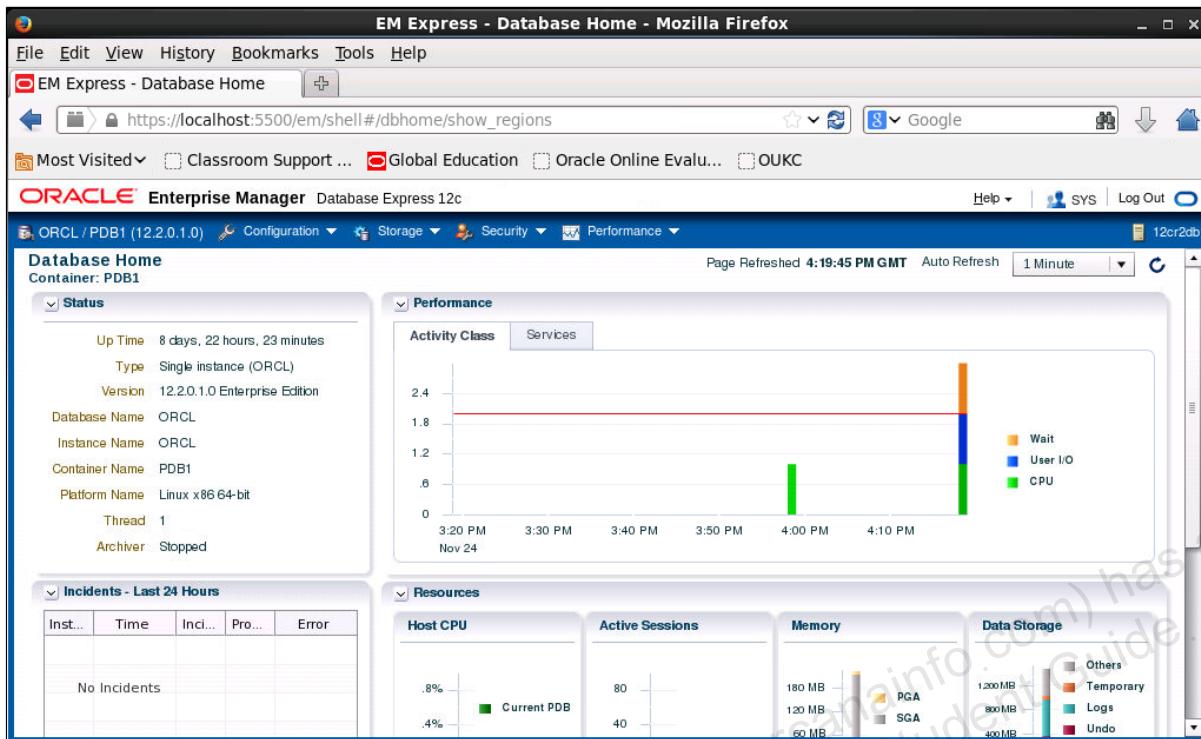


9. Click the tabs and menus to review the interface.
10. In the upper right corner, click **Log Out**.
11. Start EM Express for PDB1. To do this, on the Login page for EM Express, enter the user name **SYS** and the password as specified in [Appendix - Product-Specific Credentials](#). Enter **PDB1** as the container name. Select the **as sysdba** check box. Click **Login**.

The form contains the following fields:

- User Name: SYS
- Password: (Redacted)
- Container Name: PDB1
- as sysdba
- Login button

12. View the EM Express Home page for PDB1. Notice that in the upper left corner, the Home page identifies the container as PDB1. In the Status column you have a field named Container Name, which also indicates the container to which you are currently connected is PDB1.



13. Browse the Configuration, Storage, Security, and Performance menus.

14. Click Log Out and close the browser window.

Practice 1-4 Getting Started with SQLcl

Overview

In this practice, you launch SQLcl and using a simple query, experiment with different output formats.

Assumptions

You are currently logged in to VM1 as the `oracle` user.

Tasks

Complete the following steps on VM1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Change to the directory where the SQLcl executable resides.

```
$ cd $ORACLE_HOME/sqldeveloper/sqlcl/bin
$
```

3. List the files in the current directory.

```
$ ls
sql    sql.bat    sql.exe
$
```

4. Allow everyone to be able to read and execute the `sql` file.

```
$ chmod 755 sql
$
```

5. Make sure that you're running Java version 8 or more. Look for a 1.8 version or newer version in the results.

```
$ java -version

java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
$
```

6. Launch SQLcl and connect directly to PDB1 as the system user. See Appendix - Product-Specific Credentials for the password.

```
$ sql system/<password>@localhost:1521/PDB1.example.com

SQLcl: Release 12.2.0.1.0 RC on Tue Nov 22 20:21:53 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

Note: Alternatively, you can launch SQLcl, log in to the CDB, and then switch to PDB1:

```
$ ./sql

SQLcl: Release 12.2.0.1.0 RC on Tue Nov 22 20:21:53 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Username? (' '?)system
Password? (*****?) <password>

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL> ALTER SESSION SET CONTAINER = PDB1;

Session altered.
SQL>
```

7. Show the container name.

```
SQL> SHOW con_name

CON_NAME
-----
PDB1
SQL>
```

8. View the list of commands that you can use in SQLcl. You'll see a lot of SQL commands that are also available in SQL*Plus, plus several new ones (these are underlined and bolded).

```
SQL> help
```

For help on a topic type help <topic>
List of Help topics available:
/
@
@@
ACCEPT
ALIAS
APEX
APPEND
...
SQL>

9. Generate the DDL for the LOCATIONS table.

```
SQL> ddl hr.locations

CREATE TABLE "HR"."LOCATIONS"
(
    "LOCATION_ID" NUMBER(4,0),
    "STREET_ADDRESS" VARCHAR2(40),
...
SQL>
```

10. You can affect how the DDL is formatted by using the set ddl command. This command shows you all the different flags.

```
SQL> HELP set ddl

SET DDL [ [ PRETTY | SQLTERMINATOR | CONSTRAINTS | REF_CONSTRAINTS |
CONSTRAINTS_AS_ALTER | OID | SIZE_BYTE_KEYWORD | PARTITIONING |
...
SQL>
```

11. Create a simple query against the HR schema. By default, SQLcl uses the same column formatting as SQL*Plus. In the results below, the data is printed to the size of the column definitions.

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

FIRST_NAME          LAST_NAME          SALARY      HIRE_DATE
-----              -----              -----
Steven              King               24000      17-JUN-03
Neena               Kochhar            17000      21-SEP-05
...
107 rows selected.
SQL>
```

12. Show the SQL formatting style. Notice that the style is set to Default.

```
SQL> SHOW sqlformat
SQL Format : Default
SQL>
```

13. Change the SQL formatting style to ansiconsole.

```
SQL> SET sqlformat ansiconsole
```

14. Rerun the query against the HR.EMPLOYEES table and observe the new formatting. The columns are fit to the screen. You can use the Up arrow key to insert the SQL code.

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


| FIRST_NAME | LAST_NAME | SALARY | HIRE_DATE |
|------------|-----------|--------|-----------|
| Steven     | King      | 24000  | 17-JUN-03 |
| Neena      | Kochhar   | 17000  | 21-SEP-05 |
| ...        |           |        |           |


SQL>
```

15. You can also format the output as comma-separated values (CSV).

```
SQL> SET sqlformat csv
```

16. Enter a forward slash to rerun the previous query.

```
SQL> /
"FIRST_NAME", "LAST_NAME", "SALARY", "HIRE_DATE"
"Steven", "King", 24000, 17-JUN-03
"Neena", "Kochhar", 17000, 21-SEPT-05
...
SQL>
```

17. Hide the column headers.

```
SQL> SET head off
SQL>
```

18. Rerun the query.

```
SQL> /
"Steven", "King", 24000, 17-JUN-03
"Neena", "Kochhar", 17000, 21-SEPT-05
...
SQL>
```

19. Set the formatting back to ansiconsole, set the page size, and rerun the query. Notice that there are 50 rows per page.

```
SQL> SET sqlformat ansiconsole
SQL> set pagesize 50
SQL> /
```

FIRST NAME	LAST NAME	SALARY	HIRE DATE
Steven	King	24000	17-JUN-03
Neena	Kochhar	17000	21-SEP-05
...			
SQL>			

20. View the history of queries that you have run so far on the database. Your list may be different than the list below.

```
SQL> history

1 select name, cdb, con_id FROM v$database;
2 select name, cdb, con_id from v$database;
3 select name, cdb from v$database;
4 alter session set container = PDB1
5 ddl hr.locations
6 select first_name, last_name from hr.employees;
7 select first_name, last_name, salary, hire_date from hr.employees;
SQL>
```

21. Rerun one of the queries. In this example, query 2 is rerun. Note: Entering "h 2" will do the same thing as entering "history 2."

```
SQL> history 2

1 select name, cdb, con_id
2* from v$database

SQL> /


| <b>NAME</b> | <b>CDB</b> | <b>CON_ID</b> |
|-------------|------------|---------------|
| ORCL        | YES        | 0             |


SQL>
```

22. Exit SQLcl.

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64
bit Production
$
```

2

Managing Database Instances



ORACLE®

Objectives for Lesson 2

After completing this lesson, you should be able to:

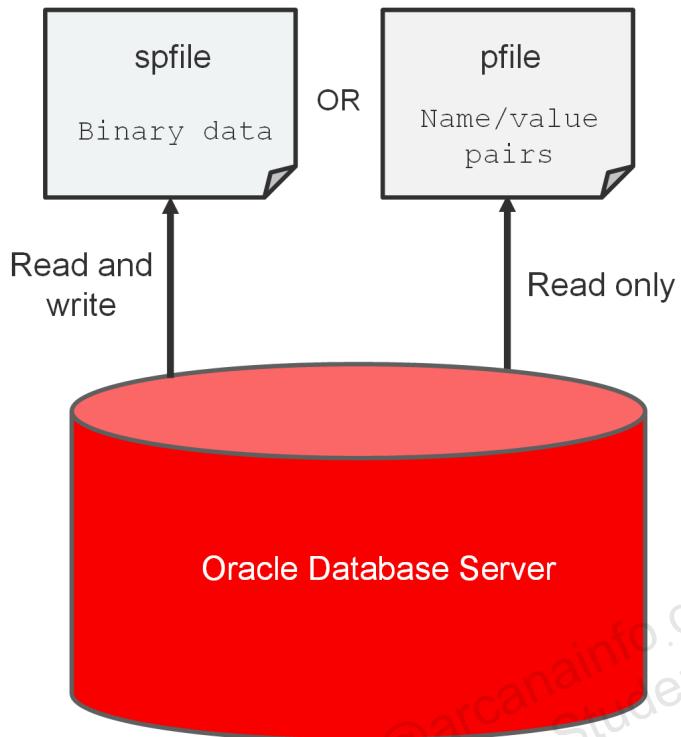
- Describe initialization parameter files and initialization parameters
- View and modify initialization parameters in SQL*Plus
- Start up and shut down Oracle Databases
- Open and close PDBs
- Work with the Automatic Diagnostic Repository (ADR)
- Query dynamic performance views



ORACLE®

Now that you've had a chance to explore a CDB and PDB, you're ready to start manipulating them. One of the most basic tasks you'll do is start up and shut down your Oracle database because if it's not started up, you can't access the data in it. So therefore, it's at this time that you need to become familiar with the initialization parameter file. It's an important file and useful to learn early on because it determines how your system starts itself up and how your system operates. Also, inside an initialization parameter file are initialization parameters, and these parameters are discussed and used throughout the course.

Working with Initialization Parameters



ORACLE

Initialization Parameter Files

When you start a database instance, it reads instance configuration parameters (initialization parameters) from an *initialization parameter file* (parameter file). On most platforms, parameter files are stored in the \$ORACLE_HOME/dbs directory by default.

You can use one of the following types of parameter files to start your database instance, as illustrated above:

- **Server parameter file (spfile):** An spfile is a binary file that is written to and read by the database server. You can't edit it manually. An spfile is preferred over a pfile because you can change initialization parameters with ALTER SYSTEM commands in SQL*Plus and the changes persist when you shut down and start up the database instance. It also provides a basis for self-tuning by Oracle Database. An spfile is automatically created for you by Database Configuration Assistant (DBCA) when you create a CDB. It resides on the server on which the Oracle instance is running. The default name of the spfile, which is automatically sought at startup, is spfile<SID>.ora.
- **Text initialization parameter file (pfile):** A pfile is a text file containing parameter values in name/value pairs, which the database server can read to start the database instance. Unlike an spfile, the database server cannot write to and alter a pfile. Therefore, to change parameter values in a pfile and make them persist during shutdown and startup, you must manually edit the pfile in a text editor and restart the database instance to refresh the parameter values. Your installation includes a sample pfile named init.ora in the default directory for parameter files. You can use this file as a starting point for a pfile, or you can create a pfile from

the spfile. If you save your pfile as `init<SID>.ora` in the default directory, the database server will automatically use it if an spfile is not available. If you save the pfile under a different name, you'll need to specify it during startup.

Search Order for a Parameter File

The database server locates your parameter file by examining file names in the `$ORACLE_HOME/dbs` directory in the following order:

1. `spfile<SID>.ora`, where `SID` is the system ID and identifies the instance name (for example, `ORCL`)
2. `spfile.ora`
3. `init<SID>.ora` (pfile)

Initialization Parameters

- *Initialization parameters* (parameters) do the following:
 - Set database limits
 - Set database-wide defaults
 - Specify files and directories
 - Affect performance
- Parameters can be of two types: basic or advanced.
 - Tune around 30 basic parameters to get reasonable database performance.
 - Example basic parameter: SGA_TARGET
 - Example advanced parameter: DB_CACHE_SIZE
- *Derived parameters* calculate their values from the values of other parameters.
 - Example: SESSIONS is derived from PROCESSES.
- Some parameter values or value ranges depend on the host operating system.
 - Example: DB_BLOCK_SIZE

ORACLE

Parameter Values That Depend on the OS

Some parameter values or value ranges depend on the host operating system. For example, the DB_FILE_MULTIBLOCK_READ_COUNT parameter specifies the maximum number of blocks that are read in one I/O operation during a sequential scan; this parameter is platform dependent. The size of those blocks, which is set by DB_BLOCK_SIZE, has a default value that depends on the operating system.

View:

Review the information about the following parameter. *Oracle Database Reference* is a good source of information about parameters. In this example, you'll learn the parameter's datatype (string), syntax (CONTROL_FILES = filename, [, filename] . . .), default value (operating system-dependent), whether its modifiable (no), whether you can modify its value in a PDB (no), its range of values (1 to 8 filenames), whether it is a basic parameter (yes), and details for Oracle RAC (multiple instances must have the same value).

- CONTROL_FILES

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Modifying Initialization Parameters

- Modify parameters to set capacity limits or improve performance.
 - Use EM Express or SQL*Plus (`ALTER SESSION` or `ALTER SYSTEM`).
- Query the `V$PARAMETER` view for a particular initialization parameter to learn whether you can make:
 - Session-level changes (`ISSES_MODIFIABLE` column)
 - System-level changes (`ISSYS_MODIFIABLE` column)
 - PDB-level changes (`ISPDB_MODIFIABLE` column)
- Use the `SCOPE` clause with the `ALTER SYSTEM` command to tell the system *where* to update the system-level parameter:
 - MEMORY
 - SPFILE
 - BOTH
 - DEFERRED

ORACLE®

Learning Before Modifying

Before modifying a parameter, you should query the V\$PARAMETER view to learn about *how* you can modify a parameter.

Information about...	Column in the V\$PARAMETER View	Notes
Session-level changes	The ISSES_MODIFIABLE column value tells you whether you can change the parameter for your current session (TRUE) or not (FALSE) by using the ALTER SESSION command.	You can change some parameters at the session level, but not all. Changes are applied to your current session immediately (<i>dynamically</i>) and expire when you end your session. Parameters with a value of TRUE are referred to as <i>session-level</i> parameters. Example: SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'mon dd yyyy';
System-level changes	The ISSYS_MODIFIABLE column value tells you when a system-level change to the parameter, made by using the ALTER SYSTEM command, takes effect. <ul style="list-style-type: none">IMMEDIATE means the change will take effect immediately and be applied to all current sessions.DEFERRED means the change will take effect in subsequent sessions.FALSE means the change will take effect in subsequent instances.	You can change all parameters at the system level by using the ALTER SYSTEM command, and the change is applied to all sessions. Parameters with a value of FALSE are referred to as <i>static</i> parameters. For static parameters, you need to shut down and restart the database instance to implement the change. Also, the database instance must have been started with an spfile. Example: SQL> ALTER SYSTEM SET SEC_MAX_FAILED_LOGIN_ATTEMPTS=2 SCOPE=SPFILE;
PDB-level changes	The ISPDB_MODIFIABLE column value tells you whether you can (TRUE) or can't (FALSE) modify the parameter inside a PDB.	In a non-CDB, the value of this column is NULL.

Setting the Scope in the ALTER SYSTEM Command

Use the SCOPE clause with the ALTER SYSTEM command to tell the system *where* to update the system-level parameter. This location dictates *how long* the change will stay in effect. Scope also depends on whether you started the database instance using a pfile or an spfile. Scope can have the following values:

- MEMORY:** This value tells the system to make the parameter change in memory only. The change will take effect immediately, but will not persist in subsequent sessions. If you started the database instance using a pfile, then this is the only scope you can specify. This specification is not allowed for static parameters.
- SPFILE:** This value tells the system to make the parameter change in the spfile only. The change will take effect immediate and will persist after you restart the database instance. This is the only scope allowed for static parameters.
- BOTH:** This value tells the system to make the parameter change in both memory and in the spfile. The change will take effect immediately and persist after you restart the database instance. If you started the database instance using an spfile, then BOTH is the default.

- **DEFERRED:** This value tells the system to make the parameter change effective only for future sessions. This specification is valid only for a small number of dynamic parameters, for example, AUDIT_FILE_DEST.

Viewing Initialization Parameters

Ways to view initialization parameters in SQL*Plus:

- Issue the `SHOW PARAMETER` command.
 - Example: Find information about all parameters whose names contain the word 'para':

```
SQL> SHOW PARAMETER para
```
- Query the following views:
 - `V$PARAMETER`
 - `V$PARAMETER2`
 - `V$SPPARAMETER`
 - `V$SYSTEM_PARAMETER`

ORACLE

Issuing the `SHOW PARAMETER` Command

You can issue the `SHOW PARAMETER` command in SQL*Plus to view information about an initialization parameter; for example, view a parameter's datatype and default value. For example, the following `SHOW PARAMETER` command returns information about parameters whose names contain the word 'para.'

```
SQL> SHOW PARAMETER para
```

NAME	TYPE	VALUE
cell_offload_parameters	string	
fast_start_parallel_rollback	string	LOW
parallel_adaptive_multi_user	boolean	TRUE
parallel_automatic_tuning	boolean	FALSE
...		

Querying Views

You can also query the `V$PARAMETER` view in SQL*Plus to view information about an initialization parameter. For example, the following query against the `V$PARAMETER` view returns information about parameters whose names contain the word 'pool.'

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

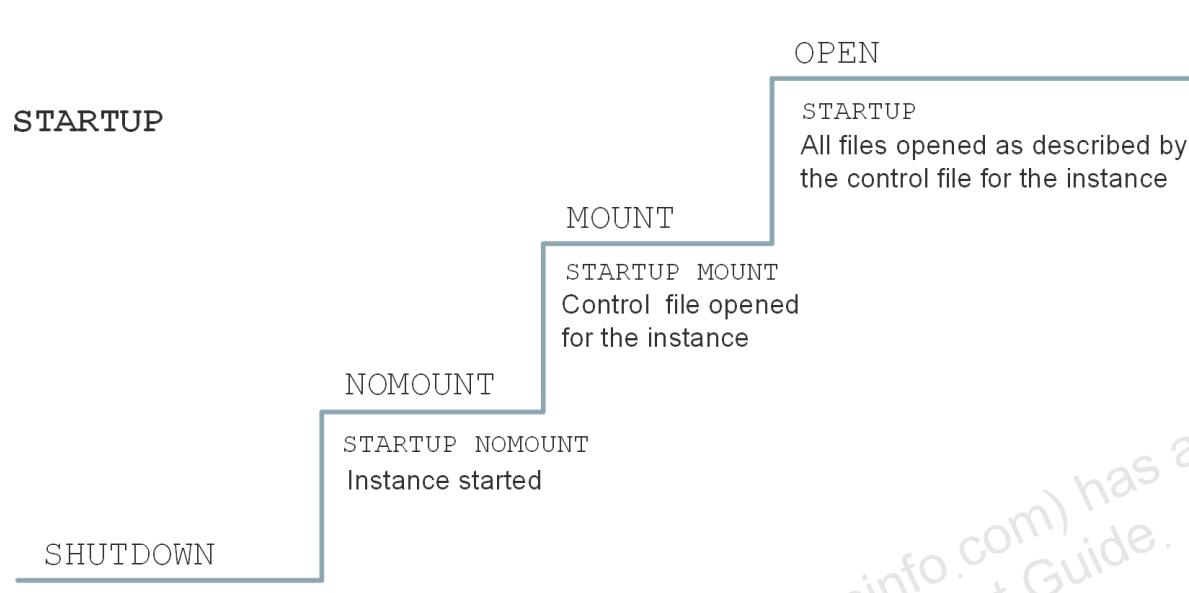
NAME	VALUE
shared_pool_size	0
large_pool_size	0
java_pool_size	0
streams_pool_size	0
shared_pool_reserved_size	15728640

9 rows selected.

Other views that contain parameter information include:

- **V\$SPPARAMETER**: Displays information about the contents of the spfile. If you didn't use an spfile to start the database instance, each row of the view will contain FALSE in the ISSPECIFIED column.
- **V\$PARAMETER2**: Displays information about the parameters that are currently in effect for the session, with each parameter value appearing as a row in the view. A new session inherits parameter values from the database instance-wide values displayed in the V\$SYSTEM_PARAMETER2 view.
- **V\$SYSTEM_PARAMETER**: Displays information about the parameters that are currently in effect for the database instance

Starting Up Oracle Databases



ORACLE®

Before users can connect to a database instance, a database administrator must start up the database server. To do this, three tasks are required (as illustrated in the diagram above): start the database instance, mount the database instance, and then open the database. You can use the `STARTUP` command in SQL*Plus to perform all three tasks at once or to perform each one separately.

1. **Starting the database instance** puts the database server in `NOMOUNT` mode. During this step, the Oracle software reads an initialization parameter file, starts background processes, allocates memory to the SGA, and opens the alert log and trace files.
2. **Mounting the database instance** puts the database server in `MOUNT` mode. During this step, the Oracle software associates the database (CDB) with the previously started database instance, opens and reads the control files that are specified in the initialization parameter file, and obtains the names and statuses of the data files and online redo log files. No checks, however, are performed to verify the existence of the data files and online redo log files at this time. Start up in `MOUNT` mode to perform some maintenance operations, such as renaming data files and performing full database recoveries.
3. **Opening the database** puts the database in `OPEN` mode. The Oracle software opens the redo log files and data files according to the list registered in the control files. Start up in `OPEN` mode to enable users to connect to the database instance. PDBs are not, by default, started up when you open the database.

Shutting Down Oracle Databases

- Sometimes you need to shut down the database instance, for example, to change a static parameter or patch the database server.
- Use the `SHUTDOWN` command to shut down the database server in various modes: `ABORT`, `IMMEDIATE`, `TRANSACTIONAL`, and `NORMAL`.

	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Allows new connections	No	No	No	No
Waits until current sessions end	No	No	No	Yes
Waits until current transactions end	No	No	Yes	Yes
Forces a checkpoint and closes files	No	Yes	Yes	Yes

ORACLE®

IMMEDIATE Mode

A shutdown in `IMMEDIATE` mode is the most typically used option.

- Current SQL statements being processed by the database instance are not completed.
- The database server does not wait for the users who are currently connected to the database instance to disconnect.
- The database server rolls back active transactions and disconnects all connected users.
- The database server closes and dismounts the database before shutting down the database instance.

TRANSACTIONAL Mode

A shutdown in `TRANSACTIONAL` mode prevents clients from losing data, including results from their current activity.

- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have been completed, a shutdown occurs immediately.

NORMAL Mode

`NORMAL` is the default shutdown mode if no mode is specified with the `SHUTDOWN` command.

- No new connections can be made.

- The Oracle server waits for all users to disconnect before completing the shutdown.
- Database and redo buffers are written to disk.
- Background processes are terminated and the SGA is removed from memory.
- The Oracle server closes and dismounts the database before shutting down the instance.

ABORT Mode

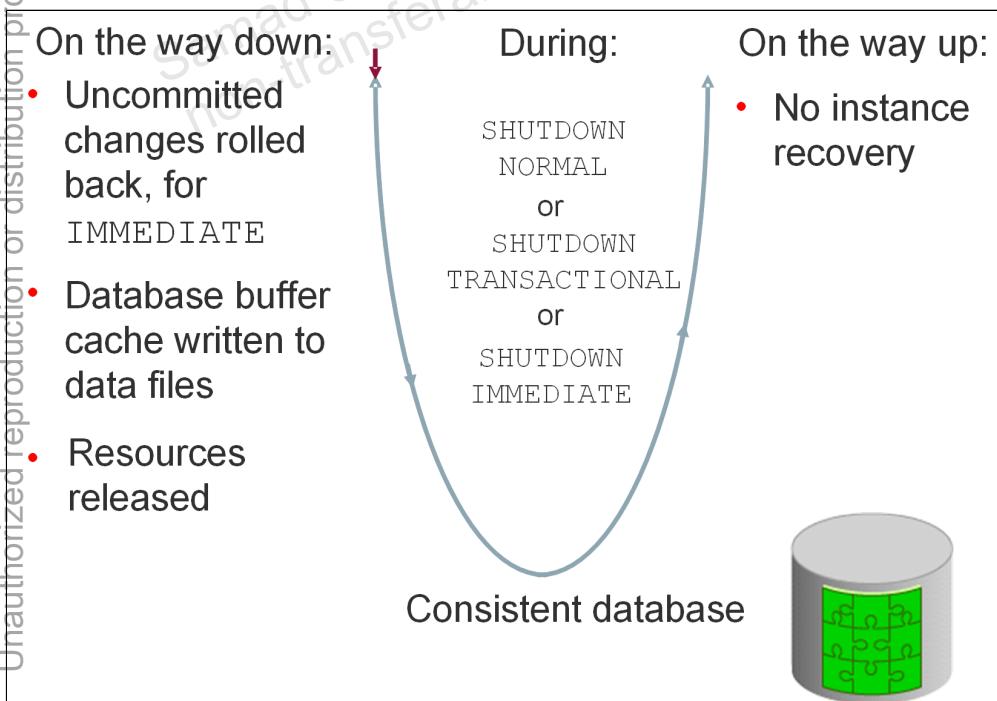
If the other shutdown modes don't work, you can use the ABORT mode. ABORT mode performs the least amount of work before shutting down. Because this mode puts the database in an inconsistent state and requires recovery before startup, use it only when necessary. It's not advisable to back up the database in this state. It's typically used when no other form of shutdown works, when there are problems with starting the database instance, or when you need to shut down immediately because of an impending situation (such as notice of a power outage within seconds). ABORT mode is usually the fastest shutdown mode and NORMAL mode is the slowest. NORMAL and TRANSACTIONAL modes can take a long time depending on the number of sessions and transactions.

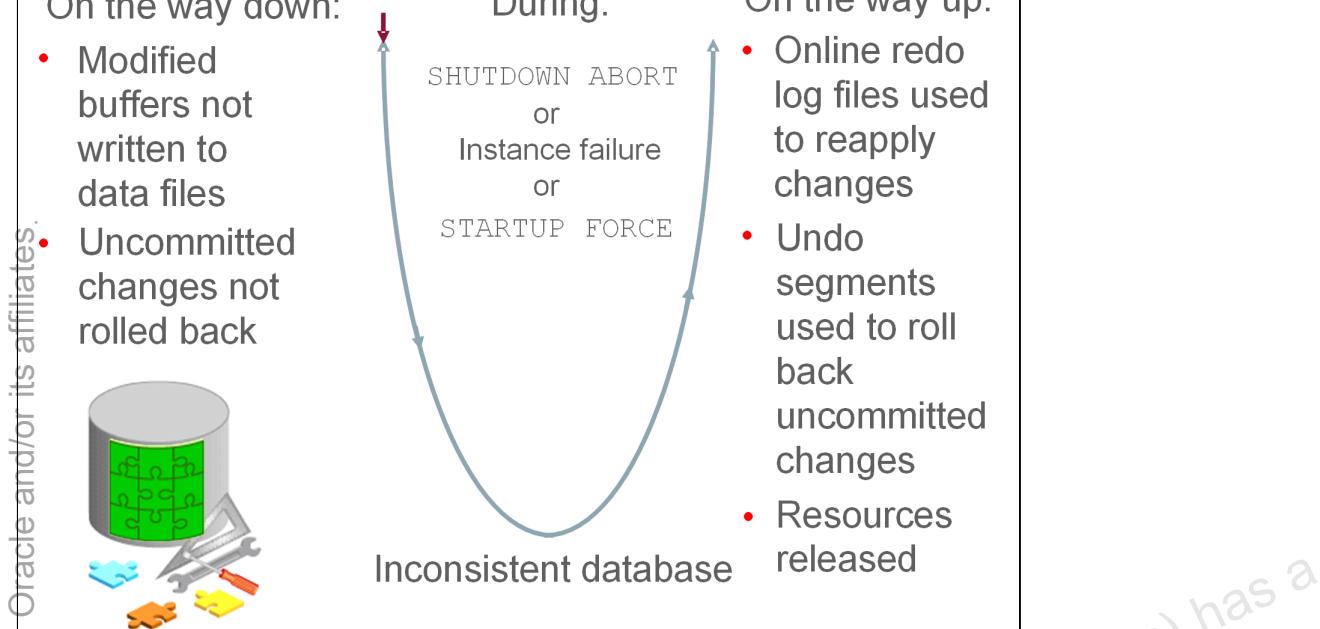
The following happens during a shutdown in ABORT mode, an instance failure, or a database instance startup in FORCE mode:

- Current SQL statements being processed by the Oracle server are immediately terminated.
- The Oracle server does not wait for users who are currently connected to the database to disconnect.
- Database and redo buffers are not written to disk.
- Uncommitted transactions are not rolled back.
- The instance is terminated without closing the files.
- The database is not closed or dismounted.
- The next startup requires instance recovery, which occurs automatically.

Comparing Shutdown Modes

The following diagrams compare the IMMEDIATE, TRANSACTIONAL, and NORMAL shutdown modes to the ABORT shutdown mode (and instance failure or STARTUP FORCE mode). Notice that the database becomes inconsistent when you perform an ABORT shutdown, whereas it stays consistent during the other shutdown modes. Also note that you need to recover the database instance after you perform an ABORT shutdown; whereas with the other shutdown modes, you don't need to do so.





Opening and Closing PDBs

- Open/close a PDB to open/close its data files.
- A PDB has four open modes:
 - READ WRITE (the PDB is fully started/opened)
 - READ ONLY
 - MIGRATE
 - MOUNTED (the PDB is shut down/closed)
- Use the ALTER PLUGGABLE DATABASE command or STARTUP and SHUTDOWN commands to open and close PDBs.
 - Example:

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

- The ALTER PLUGGABLE DATABASE command lets you change from any open mode to another.
- To use the STARTUP command, the PDB must first be in MOUNTED mode.

ORACLE

Open Modes

Starting up a PDB and opening a PDB mean the same thing, and you'll find both phrases used in documentation and online resources. When you open a PDB, the database server opens the data files for that PDB. Similar to a CDB, a PDB has four levels of being open, and these levels are referred to as open modes. The open modes are READ WRITE (the PDB is fully started/opened), READ ONLY, MIGRATE, and MOUNTED (the PDB is shut down/closed).

View:

Click the following link to review each open mode.

- [ALTER PLUGGABLE DATABASE](#) (Scroll down to Table 11-2 PDB Open Modes)

Commands to Open and Close PDBs

You can use the `ALTER PLUGGABLE DATABASE` command to open and close a PDB from either the root container or within the PDB itself. You can also use `STARTUP` and `SHUTDOWN` commands. The `ALTER PLUGGABLE DATABASE` command lets you change from any open mode to another for a PDB. To use the `STARTUP` command, the PDB must first be in `MOUNTED` mode. Either command requires you to be connected to the root container or PDB with one of the following system privileges: AS SYSBACKUP, AS SYSDBA, AS SYSDG, or AS SYSOPER.

Examples

In this example, PDB1 is started up (opened). Its open mode is changed from `MOUNT` to `READ WRITE`.

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

In this example, PDB1 is shut down (closed). Its open mode is changed to `MOUNT`.

```
SQL> ALTER PLUGGABLE DATABASE PDB1 CLOSE;
```

View:

Click the following link to view other examples using the `ALTER PLUGGABLE DATABASE` command.

- [ALTER PLUGGABLE DATABASE](#) (Scroll down to the subtopic [Changing the State of a PDB: Examples](#))

Working with the Automatic Diagnostic Repository

The Automatic diagnostic repository (ADR):

- Is a file-based repository outside the database
- Is a system-wide central tracing and logging repository
- Stores database diagnostic data such as:
 - Traces
 - Alert log
 - Health monitor reports



About the Automatic Diagnostic Repository

The ADR is a system-wide tracing and logging central repository for database diagnostic data such as traces, the alert log, health monitor reports, and more.

The ADR root directory is known as ADR base. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter, for example, `/u01/app/oracle`.

The location of an ADR home is given by the following path, which starts at the ADR base directory:

```
<ADR Base>/diag/product_type/db_id/instance_id
```

For example:

```
/u01/app/oracle/diag/rdbms/orcl/ORCL
```

Viewing the Alert Log

- The alert file is a chronological log of messages about the database instance and database, such as:
 - Any nondefault initialization parameters used at startup
 - All internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60) that occurred
 - Administrative operations, such as the SQL statements CREATE, ALTER, DROP DATABASE, and TABLESPACE; and the Enterprise Manager or SQL*Plus statements STARTUP, SHUTDOWN, ARCHIVE LOG, and RECOVER
 - Several messages and errors relating to the functions of shared server and dispatcher processes
 - Errors during the automatic refresh of a materialized view
- Query V\$DIAG_INFO to find the location of the alert log.
 - The path to alert_SID.log corresponds to the Diag Trace entry.
 - The path to log.xml corresponds to the Diag Alert entry.
- You can view the alert log in a text editor or in ADRCI.

ORACLE

Each database instance has an alert_SID.log file. The file is on the server with the database and is stored in \$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/trace by default if \$ORACLE_BASE is set.

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. Many systems also display this information on the console. If an administrative operation is successful, a message is written in the alert log as "completed" along with a time stamp.

Enterprise Manager Cloud Control monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and information messages. Because the file can grow to an unmanageable size, you can periodically back up the alert file and delete the current alert file. When the database attempts to write to the alert file again, it creates a new one.

Note: There is an XML version of the alert log in the \$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/alert 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 Automatic Diagnostic Repository (ADR) and to view the alert log. ADRCI has a rich command set that you can use interactively or in scripts.

Using Trace Files

- Trace files contain:
 - Error information (contact Oracle Support Services if internal error occurs)
 - Information that can provide guidance for tuning applications or an instance
- Each server and background process can write to an associated trace file.
- Trace file names for background processes are named after their processes.
 - Exception: Trace files generated by job queue processes
- Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems.
- When a critical error occurs:
 - An incident number is assigned to the error.
 - Diagnostic data for the error (such as trace files) is immediately captured and tagged with the incident number.
 - Data is stored in the ADR
- ADR files can be automatically purged with retention policy parameters.

ORACLE®

About Trace 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. If an internal error occurs and information is written to a trace file, the administrator should contact Oracle Support Services.

All file names of trace files associated with a background process contain the name of the process that generated the trace file. The one exception to this is trace files that are generated by job queue processes (Jnnn).

Additional information in trace files can provide guidance for tuning applications or an instance. Background processes always write this information to a trace file when appropriate.

Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems. In particular, problems that are targeted include critical errors such as those caused by database code bugs, metadata corruption, and customer data corruption.

When a critical error occurs, an incident number is assigned to it; diagnostic data for the error (such as trace files) is immediately captured and tagged with this number. The data is then stored in the automatic diagnostic repository (ADR)—a file-based repository outside the database—where it can later be retrieved by incident number and analyzed.

Purging Mechanism

The purging mechanism allows you to specify a retention policy stating:

- How old ADR contents should be before they are automatically deleted.
 - The long retention period is used for the relatively higher-value diagnostic data, such as incidents and alert log. (Default value is 365 days)
 - The short retention period is used for traces and core dumps. (Default value is 30 days).

Older items are deleted first. The long retention period items are typically older than any of the items in the short retention period. So a mechanism is used in which the time periods are “scaled,” so that roughly the same percentage of each gets deleted. Some components use these periods in slightly different ways. For instance, IPS, the packaging facility, uses the short retention period to determine when to purge packaging metadata and the staging directory contents. However, the age of the data is based on when the package was completed, not when it was originally created.

- The size-based retention to specify a target size for an ADR home. When purging, the old data, determined by the time-based retention periods, is deleted first. If the size of the ADR home is still greater than the target size, diagnostics are automatically deleted until the target size is no longer exceeded.

Administering the DDL Log File

- Enable the capture of certain DDL statements to a DDL log file by setting `ENABLE_DDL_LOGGING` to `TRUE`.
- DDL log contains one log record for each DDL statement.
- Two DDL logs containing the same information:
 - XML DDL log: `log.xml` written to
\$ORACLE_BASE/diag/rdbms/<dbname>/<SID>/log/ddl
 - Text DDL: `ddl_<sid>.log` written to
\$ORACLE_BASE/diag/rdbms/<dbname>/<SID>/log
- Example:

```
$ more ddl_orcl.log
Thu Nov 15 08:35:47 2016
diag_ad1:drop user app_user
```



The DDL log is created only if the `ENABLE_DDL_LOGGING` initialization parameter is set to `TRUE`. When this parameter is set to `FALSE`, DDL statements are not included in any log. A subset of executed DDL statements is written to the DDL log.

There are two DDL logs that contain the same information. One is an XML file, and the other is a text file. The DDL log is stored in the `log/ddl` subdirectory of the ADR home.

Important: You must have a license for Database Lifecycle Management Pack to enable DDL logging.

View:

Click the following link to *Oracle Database Reference* and view the complete list of DDL commands that are captured in the DDL log.

- [ENABLE_DDL_LOGGING](#)

Querying Dynamic Performance Views

- Dynamic performance views contain information about the changing states of the instance memory structures:
 - Sessions, file states, and locks
 - Progress of jobs and tasks
 - Backup status, memory usage and allocation
 - System and session parameters
 - SQL execution
 - Statistics and metrics
- Dynamic performance views start with the prefix V\$.
- Example query: Which current sessions have logged in from the EDXX9P1 computer in the last day?

```
SELECT * FROM V$SESSION
WHERE machine = 'EDXX9P1'
AND logon_time > SYSDATE - 1;
```

ORACLE

Considerations for Dynamic Performance Views

- These views are owned by the SYS user.
- Different views are available at different times:
 - The instance has been started.
 - The database is mounted.
 - The database is open.
- You can query V\$FIXED_TABLE to see all the view names.
- These views are often referred to as “v-dollar views.”
- Read consistency is not guaranteed on these views because the data is dynamic.

ORACLE®

Summary for Lesson 2

Unauthorized reproduction or distribution prohibited
Copyright © 2017, Oracle and/or its affiliates.

In this lesson, you should have learned how to:

- Describe initialization parameter files and initialization parameters
- View and modify initialization parameters in SQL*Plus
- Start up and shut down Oracle Databases
- Open and close PDBs
- Work with the Automatic Diagnostic Repository (ADR)
- Query dynamic performance views



ORACLE

Samad Ul Haq (samad.ulhaq@arcanainformatics.com)
non-transferable license to use this material

Practice 2 Overview

- 2-1: Creating a PFILE from an SPFILE
- 2-2: Viewing Initialization Parameters with SQL*Plus
- 2-3: Modifying Initialization Parameters with SQL*Plus
- 2-4: Modifying an Initialization Parameter with EM Express
- 2-5: Shutting Down and Starting Up the Oracle Database
- 2-6: Viewing Diagnostic Information

Practice 2-1 Creating a PFILE from an SPFILE

Overview

In this practice, you create a pfile named `initORCL.ora` from the default spfile while being connected to the root container of your CDB.

Unauthorized reproduction or distribution prohibited. Copyright © 2017, Oracle and/or its affiliates.

Tip:

When you create a pfile in a CDB, the current container can be the root or a PDB. If the current container is the root, then the database creates a text file that contains the parameter settings for the root. If the current container is a PDB, the database creates a text file that contains the parameter settings for the PDB. You must specify a `pfile_name`, and you must have the `SYSDBA`, `SYSOPER`, or `SYSBACKUP` system privilege to execute the `CREATE PFILE` statement. You can execute this statement either before or after you start a database instance.

The name of the parameter file varies depending on your operating system. For example, it may be in mixed case or lowercase, or it may have a logical name or a variation on the name `INIT.ORA`. As the DBA, you can choose a different filename for your parameter file.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Complete the following steps on VM1. Note: The results for some steps are formatted for easier viewing.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

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

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

3. Locate the default spfile for your database instance. The results show that the spfile is located at `/u01/app/oracle/product/12.2.0/dbhome_1/dbs/spfileORCL.ora`, which is `$ORACLE_HOME/dbs`. Minimize the terminal window for a moment.

```
SQL> SHOW PARAMETER spfile
NAME          TYPE        VALUE
-----
spfile        string      /u01/app/oracle/product/12.2.0
                           /dbhome_1/dbs/spfileORCL.ora
SQL>
```

4. On your desktop, open **Computer**, and then **Filesystem**. Browse to the `$ORACLE_HOME/dbs` directory (`/u01/app/oracle/product/12.2.0/dbhome_1/dbs`). Notice that the spfile (`spfileORCL.ora`) and `init.ora` files are stored here. The naming convention for an spfile is `spfile<SID>.ora`.
5. View the `init.ora` file. This is a sample pfile provided with the Oracle Database installation.
 - a. Right-click `init.ora`, and select **Rename**.
 - b. Change the name to `init.ora.txt`.
 - c. Double-click `init.ora.txt` to open it in gedit.
 - d. Scroll through the file (the code is shown below) to view the parameters.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

```
#####
# 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'
```

- e. Close the file and rename it back to init.ora.
6. In the terminal window, create a pfile named **initORCL.ora** from the default spfile by using the **CREATE PFILE** statement. By naming the file as **initORCL.ora**, you put the file in the default search order for parameter files. If the database server doesn't find an spfile, then this file will be used.
 A few things to know about the **CREATE PFILE** statement: If you do not specify a name for the pfile, then the database uses the platform-specific default initialization parameter file name. If you do not specify a path to the pfile, then the database adds the path for the default storage location, which is platform dependent. You can specify the name of the spfile in the default directory, or specify the full path to an spfile located somewhere else. If you don't specify any name, the database will use the spfile name currently associated with the database instance. If no spfile is associated with the instance, then the database looks for the platform-specific default server parameter file name. If that file does not exist, then the database returns an error.

```
SQL> CREATE PFILE = 'initORCL.ora' FROM SPFILE;
File created.
SQL>
```

7. Verify that the pfile you just created is listed in the \$ORACLE_HOME/dbs directory and view the file.
- In the "dbs - File Browser" window, verify that the pfile named `initORCL.ora` is listed.
 - Right-click `initORCL.ora`, and select **Rename**.
 - Rename the file as `init.ora.txt`.
 - Double-click `init.ora.txt` to open it in gedit.
 - Scroll through the file (the code is shown below) to view the parameters. Because you created the pfile when you were connected to the root container, the pfile contains the parameter settings for the root container only.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

```
ORCL.__data_transfer_cache_size=0
ORCL.__db_cache_size=1157627904
ORCL.__inmemory_ext_roarea=0
ORCL.__inmemory_ext_rwarea=0
ORCL.__java_pool_size=16777216
ORCL.__large_pool_size=33554432
ORCL.__oracle_base='/u01/app/oracle'#ORACLE_BASE set from environment

ORCL.__pga_aggregate_target=822083584
ORCL.__sga_target=2466250752
ORCL.__shared_io_pool_size=134217728
ORCL.__shared_pool_size=1107296256
ORCL.__streams_pool_size=0
*.audit_file_dest='/u01/app/oracle/admin/ORCL/adump'
*.audit_trail='db'
*.compatible='12.2.0'
*.control_files='/u01/app/oracle/oradata/ORCL/control01.ctl','/u01/
app/oracle/fast_recovery_area/ORCL/control02.ctl'
*.db_block_size=8192
*.db_domain='example.com'
*.db_name='ORCL'
*.db_recovery_file_dest='/u01/app/oracle/fast_recovery_area/ORCL'
*.db_recovery_file_dest_size=12780m
*.diagnostic_dest='/u01/app/oracle'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'
*.enable_pluggable_database=true
*.local_listener='LISTENER_ORCL'
*.nls_language='AMERICAN'
*.nls_territory='AMERICA'
*.open_cursors=300
*.pga_aggregate_target=783m
*.processes=300
*.remote_login_passwordfile='EXCLUSIVE'
*.sga_target=2348m
*.undo_tablespace='UNDOTBS1'
```

- f. Close the file and rename it back to **initORCL.ora**.
8. In the terminal window, shut down the database instance in **IMMEDIATE** mode. It takes about 35 seconds. Shutting down the database instance in immediate mode causes the following things to happen:
- Current SQL statements being processed by the database instance are not completed.
 - The database server does not wait for the users who are currently connected to the database instance to disconnect.
 - The database server rolls back active transactions and disconnects all connected users.
 - The database server closes and dismounts the database before shutting down the database instance.

```
SQL> SHUTDOWN IMMEDIATE

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

9. In the "dbs - File Browser" window, in the \$ORACLE_HOME/dbs directory, rename the spfileORCL.ora file as **orig_spfileORCL.ora**. Renaming this file will take it out of the search order for parameter files when you start up the database instance. Instead, the database server will automatically find the initORCL.ora file (pfile) to start the database instance.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

10. In the terminal window, start the database instance by using the `STARTUP` command. It takes about 30 seconds.

```
SQL> STARTUP

ORACLE instance started.
Total System Global Area 2466250752 bytes
Fixed Size          8795760 bytes
Variable Size       671091088 bytes
Database Buffers   1778384896 bytes
Redo Buffers        7979008 bytes
Database mounted.
Database opened.
SQL>
```

11. 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>
```

12. Configure the database instance to start with the spfile again.

- a. Shut down the database instance in `IMMEDIATE` mode. It takes about 35 seconds.

```
SQL> SHUTDOWN IMMEDIATE

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

- b. In the "dbs - File Browser" window, in the `$ORACLE_HOME/dbs` directory, rename the `orig_spfileORCL.ora` file as `spfileORCL.ora`.
- c. In the terminal window, start the database instance by using the `STARTUP` command. It takes about 30 seconds.

```
SQL> STARTUP

ORACLE instance started.
Total System Global Area 2466250752 bytes
Fixed Size          8795760 bytes
Variable Size       671091088 bytes
Database Buffers   1778384896 bytes
Redo Buffers        7979008 bytes
Database mounted.
Database opened.
SQL>
```

- d. Verify that the database instance was started with the spfile.

```
SQL> SHOW PARAMETER spfile

NAME          TYPE        VALUE
-----
spfile        string     /u01/app/oracle/product/12.2.0
                           /dbhome_1/dbs/spfileORCL.ora
SQL>
```

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

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

14. Close the dbs - File Browser window.

*Samad Ullah (samad.ullah@arcanainfo.com) has a
non-transferable license to use this Student Guide.*

Practice 2-2 Viewing Initialization Parameters

Overview

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

- By using the `SHOW PARAMETER` command
- By querying the following views:
 - `V$PARAMETER`
 - `V$SPPARAMETER`
 - `V$PARAMETER2`
 - `V$SYSTEM_PARAMETER`

Tip:

In almost all dictionary views, the data is converted to uppercase. The `V$PARAMETER` view, however, is an exception - the data is stored in lowercase. Therefore, when writing SQL statements, make sure you use lowercase when enclosing text in quotation marks, for example: `WHERE parameter_name = 'job_queue_processes'`.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

[View Basic Parameters](#)

In this section, you view basic parameters with the `SHOW PARAMETER` command.

You can use the `SHOW PARAMETER` command with any string to view parameters that contain that string. For example, the command `SHOW PARAMETER processes` will return all initialization parameters that contain the word "processes." Basic parameters are those parameters that you are likely to modify.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

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

```
$ sqlplus / as sysdba
```

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:

Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>

3. View the `DB_NAME` and `DB_DOMAIN` parameters. Together, these values equal the global database name, which is `ORCL.example.com`.

- a. View the `DB_NAME` parameter. This parameter specifies the current database identifier of up to 8 characters. If you have multiple databases, the value of this parameter should match the Oracle instance identifier of each one to avoid confusion with other databases running on the system. The value of `DB_NAME` should be the same in both the standby and production initialization parameter files.

```
SQL> SHOW PARAMETER db_name
```

NAME	TYPE	VALUE
db_name	string	ORCL

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

```
SQL> SHOW PARAMETER db_domain
```

NAME	TYPE	VALUE
db_domain	string	example.com

4. View the `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` parameters. These parameters set the fast recovery area and 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. Specifying this parameter without also specifying the `DB_RECOVERY_FILE_DEST_SIZE` initialization parameter is not allowed.
- 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/ORCL
db_recovery_file_dest_size	big integer	12780M

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 will automatically distribute memory among the various SGA memory pools (buffer cache, shared pool, large pool, java pool, and streams pool), ensuring the most effective memory utilization. Note, the log buffer pool, other buffer caches (such as `KEEP` and `RECYCLE`), other block sizes, fixed SGA, and other internal allocations must be manually sized and are not affected by ASMM. The memory allocated to these pools is deducted from the total available memory for `SGA_TARGET` when ASMM is enabled.
- The manageability monitor process (MMON) computes the values of the automatically tuned memory pools to support ASMM.
- In addition to `SGA_TARGET` and `SGA_MAX_SIZE`, you can set minimum nonzero values for each memory pool if an application component needs a minimum amount of memory to function properly. ASMM will treat those values as minimum levels.
- The range of values for `SGA_TARGET` can be from 64 MB to an operating system-dependent value. You can't modify this value in a PDB. The current value for this parameter is 2352 MB.
- The current `SGA_MAX_SIZE` value is the value that was accepted during the creation of the CDB, which is 2352 MB.

```
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	2352M
sga_min_size	big integer	0
sga_target	big integer	2352M
unified_audit_sga_queue_size	integer	1048576

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

```
SQL> SHOW PARAMETER undo_tablespace
```

NAME	TYPE	VALUE
undo_tablespace	string	UNDOTBS1

7. View the `COMPATIBLE` parameter. This parameter specifies the release with which Oracle must maintain compatibility. It enables you to use a new release of Oracle, while at the same time guaranteeing backward compatibility with an earlier release. This is helpful if it becomes necessary to revert to the earlier release. By default, the value for the compatible entry for this parameter is equal to the version of the Oracle Database that you have installed. In your environment, this value is 12.2.0.

```
SQL> SHOW PARAMETER compatible
```

NAME	TYPE	VALUE
compatible	string	12.2.0
noncdb_compatible	boolean	FALSE

8. View the `CONTROL_FILES` initialization parameter. This parameter specifies one or more control files, separated by commas, and including paths. One to eight file names are listed. Oracle strongly recommends that you multiplex and mirror control files.

```
SQL> SHOW PARAMETER control_files
```

NAME	TYPE	VALUE
control_files	string	/u01/app/oracle/oradata/ORCL/control01.ctl, /u01/app/oracle/fast_recovery_area/ORCL/control02.ctl

9. View the `PROCESSES`, `SESSIONS`, and `TRANSACTIONS` initialization parameters.

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

```
SQL> SHOW PARAMETER processes
```

NAME	TYPE	VALUE
aq_tm_processes	integer	1
asm_io_processes	integer	20
db_writer_processes	integer	1
gcs_server_processes	integer	0
global_txn_processes	integer	1
job_queue_processes	integer	4000
log_archive_max_processes	integer	4
processes	integer	300

- 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	

- c. View the TRANSACTIONS parameter. This parameter specifies how many rollback segments to bring online when the UNDO_MANAGEMENT initialization parameter is equal to MANUAL. A transaction is assigned to a rollback segment when the transaction starts, and it can't change for the life of the transaction. A transaction table exists in the rollback segment header with limited space, limiting how many transactions a single segment can support. Therefore X number of concurrent transactions require at least X number of rollback segments. With Oracle Automatic Undo Management, the database creates rollback segments, brings them online, takes them offline, and drops them as needed. In the past, a database administrator had to do all of that.

```
SQL> SHOW PARAMETER transactions
```

NAME	TYPE	VALUE
transactions	integer	519
transactions_per_rollback_segment	integer	5

10. 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. The

default value is 200.

```
SQL> SHOW PARAMETER db_files
```

NAME	TYPE	VALUE
db_files	integer	200

View Advanced Parameters with the SHOW PARAMETER Command

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

```
SQL> SHOW PARAMETER commit_logging
```

NAME	TYPE	VALUE
commit_logging	string	

2. 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	

3. 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, which is the case below. Otherwise, the value is 128 MB for a 64-bit platform or 48 MB for a 32-bit platform.

```
SQL> SHOW PARAMETER shared_pool_size
```

NAME	TYPE	VALUE
shared_pool_size	big integer	0

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

```
SQL> SHOW PARAMETER db_block_size
```

NAME	TYPE	VALUE
db_block_size	integer	8192

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

```
SQL> SHOW PARAMETER db_cache_size
```

NAME	TYPE	VALUE
db_cache_size	big integer	0

6. View the `UNDO_MANAGEMENT` parameter. This parameter specifies the undo space management mode that the system should use. When set to `AUTO`, the instance is started in automatic undo management mode. Otherwise, it is started in rollback undo mode. In rollback undo mode, undo space is allocated as rollback segments. In automatic undo mode, undo space is allocated as undo tablespaces. The range of values is `AUTO` or `MANUAL`. If the `UNDO_MANAGEMENT` parameter is omitted when the instance is started, the default value `AUTO` is used. This is the case on VM1.

```
SQL> SHOW PARAMETER undo_management
```

NAME	TYPE	VALUE
undo_management	string	AUTO

7. View the `MEMORY_TARGET` and `MEMORY_MAX_TARGET` parameters. `MEMORY_TARGET` specifies the Oracle system-wide usable memory. The database 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`. Vice versa, 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 then 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

- b. View the MEMORY_MAX_TARGET parameter.

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

8. View the PGA_AGGREGATE_TARGET parameter. This parameter specifies the amount of Program Global Area (PGA) memory available to all server processes attached to the database instance. This memory does not reside in the System Global Area (SGA). The database uses this parameter as a target amount of PGA memory to use. When setting this parameter, subtract the SGA from the total memory on the system that is available to the Oracle instance. The minimum value is 10 MB and the maximum value is 4096 GB minus 1. The default value is 10 MB or 20% of the size of the SGA, whichever is greater. On VM1, as shown below, the value is 783 MB.

```
SQL> SHOW PARAMETER pga_aggregate_target
NAME                      TYPE        VALUE
-----                    -----      -----
pga_aggregate_target       big integer 783M
SQL>
```

Query Views for Parameter Values

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

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

TABLE_NAME
-----
USER_ADVISOR_EXEC_PARAMETERS
USER_ADVISOR_PARAMETERS
USER_ADVISOR_SQLW_PARAMETERS
USER_XS_ACL_PARAMETERS
ALL_APPLY_PARAMETERS
ALL_CAPTURE_PARAMETERS
ALL_XS_ACL_PARAMETERS
DBA_ADVISOR_DEF_PARAMETERS
DBA_ADVISOR_EXEC_PARAMETERS
DBA_ADVISOR_PARAMETERS
DBA_ADVISOR_PARAMETERS_PROJ
DBA_ADVISOR_SQLW_PARAMETERS
DBA_APPLY_PARAMETERS
DBA_CAPTURE_PARAMETERS
DBA_HANG_MANAGER_PARAMETERS
DBA_HIST_MVPARAMETER
DBA_HIST_PARAMETER
```

```
DBA_HIST_PARAMETER_NAME
DBA_ILMPARAMETERS
DBA_LOGSTDBY_PARAMETERS
DBA_ROLLING_PARAMETERS
DBA_TSDP_POLICY_PARAMETER
DBA_XS_ACL_PARAMETERS
CDB_ADVISOR_DEF_PARAMETERS
CDB_ADVISOR_EXEC_PARAMETERS
CDB_ADVISOR_PARAMETERS
CDB_ADVISOR_PARAMETERS_PROJ
CDB_ADVISOR_SQLW_PARAMETERS
CDB_APPLY_PARAMETERS
CDB_CAPTURE_PARAMETERS
CDB_HANG_MANAGER_PARAMETERS
CDB_HIST_MVPARAMETER
CDB_HIST_PARAMETER
CDB_HIST_PARAMETER_NAME
CDB_ILMPARAMETERS
CDB_LOGSTDBY_PARAMETERS
CDB_ROLLING_PARAMETERS
CDB_TSDP_POLICY_PARAMETER
CDB_XS_ACL_PARAMETERS
NLS_DATABASE_PARAMETERS
NLS_INSTANCE_PARAMETERS
NLS_SESSION_PARAMETERS
GV$HS_PARAMETER
GV$LOGMNR_PARAMETERS
GV$NLS_PARAMETERS
GV$OBSOLETE_PARAMETER
GV$PARAMETER
GV$PARAMETER2
GV$PARAMETER_VALID_VALUES
GV$SPPARAMETER
GV$SYSTEM_PARAMETER
GV$SYSTEM_PARAMETER2
GV$SYSTEM_RESET_PARAMETER
GV$SYSTEM_RESET_PARAMETER2
V$HS_PARAMETER
V$LOGMNR_PARAMETERS
V$NLS_PARAMETERS
V$OBSOLETE_PARAMETER
V$PARAMETER
V$PARAMETER2
V$PARAMETER_VALID_VALUES
V$SPPARAMETER
V$SYSTEM_PARAMETER
V$SYSTEM_PARAMETER2
V$SYSTEM_RESET_PARAMETER
V$SYSTEM_RESET_PARAMETER2

66 rows selected.
SQL>
```

2. Explore the `V$PARAMETER` view. This view displays the current parameter values in the current session.

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

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

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

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

Below are two examples from the results. The `TRANSACTIONS` parameter is static because its `ISSYS_MODIFIABLE` value is `FALSE`, and the `PLSQL_WARNINGS` parameter is dynamic because its `ISSYS_MODIFIABLE` value is `IMMEDIATE`.

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

SQL> SELECT name, issys_modifiable, value FROM v\$parameter;		
<code>...</code>		
NAME	ISSYS_MOD	VALUE
transactions	FALSE	519
plsql_warnings	IMMEDIATE	DISABLE:ALL
<code>...</code>		
417 rows selected.		
SQL>		

- c. Query the V\$PARAMETER view again, but this time, be more specific. Include a WHERE clause to specify

all parameters that contain the word "pool". The query returns eight parameters that contain the word "pool".

```
SQL> COLUMN name FORMAT A30
SQL> COLUMN value FORMAT A10
SQL> SELECT name, value FROM v$parameter 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             25165824
buffer_pool_keep                     
buffer_pool_recycle                  
olap_page_pool_size                  0

8 rows selected.
SQL>
```

3. Explore the V\$SPPARAMETER view.

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

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

```
SQL> DESCRIBE v$spparameter
Name                                         Null?    Type
-----
FAMILY                                      VARCHAR2(80)
SID                                           VARCHAR2(80)
NAME                                          VARCHAR2(80)
TYPE                                         VARCHAR2(11)
VALUE                                         VARCHAR2(255)
DISPLAY_VALUE                               VARCHAR2(255)
ISSPECIFIED                                 VARCHAR2(6)
ORDINAL                                      NUMBER
UPDATE_COMMENT                             VARCHAR2(255)
CON_ID                                       NUMBER
SQL>
```

- Query NAME and VALUE in the V\$SPPARAMETER view. Browse the 418 rows returned by the query. The results below have been formatted for easier viewing and show only a very small portion of the results.

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

NAME                                VALUE
-----
...
instance_type
uniform_log_timestamp_format
nls_language                           AMERICAN
...
418 rows selected.

SQL>
```

4. Explore the `V$PARAMETER2` view.

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

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

```
SQL> DESCRIBE v$parameter2

Name                                Null?    Type
-----
NUM                                 NUMBER
NAME                               VARCHAR2( 80 )
TYPE                               NUMBER
VALUE                             VARCHAR2( 4000 )
DISPLAY_VALUE                     VARCHAR2( 4000 )
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>
```

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

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

NAME                                VALUE
-----
...
enable_pdb_isolation                FALSE
cdb_cluster                          FALSE
cdb_cluster_name                     ORCL
...
419 rows selected.
SQL>
```

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

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

```
SQL> DESCRIBE v$system_parameter

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

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

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

NAME                                VALUE
-----
allow_group_access_to_sga          FALSE
sga_min_size                        0
shared_pool_size                   0
...
427 rows selected.

SQL>
```

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

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 2-3 Modifying Initialization Parameters

Overview

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

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

Note: Some of the query results below have been formatted for easier viewing.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Modify a Session-Level Parameter

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

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

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

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

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

3. 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.

The query results tell us the following things:

- The `ISSES_MODIFIABLE` value is `TRUE`, which means the parameter is a session-level parameter, and you can dynamically change its value just for your session.

- The ISSYS_MODIFIABLE value is FALSE, which means the parameter is also a static-level parameter. This means that the database instance must have been started with an spfile for you to change the parameter value for all sessions, and you need to restart the database instance to implement the change.
- The ISPDB_MODIFIABLE value is TRUE, which means that you can give each PDB a different parameter value.
- The parameter's current value is null, which means the parameter is using the instance-level value. The instance-level value is determined by the NLS_TERRITORY parameter.

```
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. The value is AMERICA, which tells us that the default date format is dd-mon-rr. AMERICA also determines the local currency symbol, first day of the week, and many other things.

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

NAME	VALUE
nls_territory	AMERICA

SQL>

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

- Switch to PDB1 by using the ALTER SESSION command.

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

Session altered.

SQL>

- Open PDB1.

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

Pluggable database altered.

SQL>

- Query the last_name and hire_date columns in the employees table. This table is part of the HR sample schema. Notice the date format is dd-mon-rr.

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

LAST_NAME          HIRE_DATE
-----
King                17-JUN-03
Kochhar              21-SEP-05
De Haan              13-JAN-01
...
107 rows selected.
SQL>
```

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

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

Session altered.
SQL>
```

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

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

LAST_NAME          HIRE_DATE
-----
King                jun 17 2003
Kochhar              sep 21 2005
De Haan              jan 13 2001
...
107 rows selected.
SQL>
```

8. 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>
```

9. Disconnect from PDB1 to end your session.

```
SQL> DISCONNECT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

10. What do you think happened when you ended your session?

Answer: If the NLS_DATE_FORMAT parameter is a session-level parameter, the date format should have reverted back to the default format for the system, which is 'dd-mon-rr.'

11. Connect to PDB1 again as the SYSTEM user by using the Easy Connect syntax. See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CONNECT system/<password>@localhost:1521/PDB1.example.com
```

```
Connected.
```

```
SQL>
```

12. Rerun the query against the EMPLOYEES table. The date format has reverted back to the default format dd-mon-rr. This is how a session-level parameter change works. The change only lasts for the duration of the session.

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

LAST_NAME	HIRE_DATE
King	17-JUN-03
Kochhar	21-SEP-05
De Haan	13-JAN-01
...	

```
SQL>
```

13. 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>
```

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

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$ 

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

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

The query results tell us the following things:

- The ISSES_MODIFIABLE value is FALSE, which means the parameter is not a session-level parameter. You can't change the parameter's value dynamically for just your session by using the ALTER SESSION command.
- The ISSYS_MODIFIABLE value is IMMEDIATE, which means the parameter is a dynamic, system-level parameter. You can change the parameter's value dynamically for all sessions by using the ALTER SYSTEM command.
- The parameter's current value is 4000.

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

NAME	ISSES	ISSYS_MOD	VALUE
job_queue_processes	FALSE	IMMEDIATE	4000

3. Change the JOB_QUEUE_PROCESSES parameter to 15 by using the ALTER SYSTEM command. How long the change stays in effect depends on the how you set the scope. In this example, set SCOPE equal to BOTH so that the change happens in both the database instance memory (which makes the change immediate) and in the spfile (which makes the change permanent).

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

System altered.

```
SQL>
```

4. Use the SHOW PARAMETER command to verify that the JOB_QUEUE_PROCESSES parameter value is now equal to 15.

Notice that job was entered in 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 will find all parameters that contain the letters job. In this example, the database found two parameters that contain the letters job: job_queue_processes and max_datapump_jobs_per_pdb.

The query result indicates that the job_queue_processes value is changed to 15.

```
SQL> SHOW PARAMETER job

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

Note: In Practice 2-4 Modifying an Initialization Parameter with EM Express, you change the job_queue_processes parameter value back to 4000.

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

- a. Shut down the database instance with the immediate mode. It takes about 35 seconds.

Note: In Practice 2-5 Shutting Down and Starting Up the Oracle Database, you'll look more closely at shutting down and starting up a database instance.

```
SQL> SHUTDOWN IMMEDIATE

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

- b. Start the database instance by using the STARTUP command. It takes about 30 seconds.

```
SQL> STARTUP

ORACLE instance started.
Total System Global Area 2466250752 bytes
Fixed Size                  8795760 bytes
Variable Size                671091088 bytes
Database Buffers            1778384896 bytes
Redo Buffers                 7979008 bytes
Database mounted.
Database opened.
SQL>
```

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

```
SQL> SHOW PARAMETER job
```

NAME	TYPE	VALUE
job_queue_processes	integer	15
max_datapump_jobs_per_pdb	integer	100
SQL>		

Modify a Static System-Level Parameter

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

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

The query results tell us the following things:

- The `ISSES_MODIFIABLE` value is `FALSE`, which means the parameter is not a session-level parameter. You cannot change the parameter's value dynamically for just your session by using the `ALTER SESSION` command.
- The `ISSYS_MODIFIABLE` value is `FALSE`, which means the parameter is a static parameter. You can change the parameter for all sessions by using the `ALTER SYSTEM` command only if you had started the instance with an spfile (which you did). You also have to restart the database instance for your change to take effect.
- The parameter's current value is 3.

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

NAME	ISSES	ISSYS_MOD	VALUE
sec_max_failed_login_attempts	FALSE	FALSE	3
SQL>			

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

Note: If you copy/paste this command from another source, for example, a PDF file, be sure to not break up the text within a clause; otherwise, you will get an error.

```
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 three because you need to restart the database instance for the change to take effect, which is required for static parameters.

NAME	TYPE	VALUE
sec_max_failed_login_attempts	integer	3

4. Restart the database and then verify that the new value for the SEC_MAX_FAILED_LOGIN_ATTEMPTS parameter is updated.

- a. Shut down the database instance with the immediate mode. It takes about 35 seconds.

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

- b. Start the database instance by using the STARTUP command. It takes about 30 seconds.

```
SQL> STARTUP
ORACLE instance started.
Total System Global Area 2466250752 bytes
Fixed Size                  8795760 bytes
Variable Size                671091088 bytes
Database Buffers             1778384896 bytes
Redo Buffers                 7979008 bytes
Database mounted.
Database opened.
SQL>
```

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

```
SQL> SHOW PARAMETER sec_max

NAME                      TYPE        VALUE
-----
sec_max_failed_login_attempts      integer      2
SQL>
```

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

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

NAME                      UPDATE_COMMENT
-----
sec_max_failed_login_attempts      Reduce for tighter security.
SQL>
```

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

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

Practice 2-4 Modifying an Initialization Parameter with EM Express

Overview

In this practice, you modify the `JOB_QUEUE_PROCESSES` initialization parameter (parameter) with Oracle Enterprise Manager Express (EM Express). In the previous practice, [Practice 2-3 Modifying Initialization Parameters](#), you changed this parameter value to 15. In this practice, you change its value back to 4000.

Assumptions

You are logged in to VM1 as the `oracle` user.

You completed [Practice 2-3 Modifying Initialization Parameters](#).

Tasks

Complete the following steps on VM1.

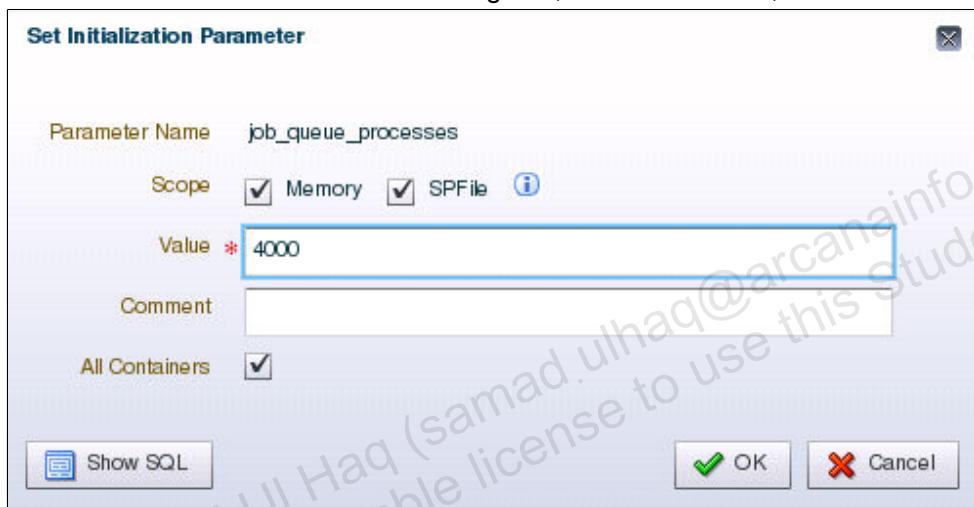
1. Open Firefox and enter the URL `https://localhost:5500/em` to connect to EM Express.
2. On the Login page, do the following, and click **Login**.
 - a. In the User Name box, enter `sys`.
 - b. In the Password box, enter the password for the `sys` user, as specified in [Appendix - Product-Specific Credentials](#).
 - c. Leave the Container Name box empty.
 - d. Select **as sysdba**.
3. On the Home page, select **Configuration**, and then **Initialization Parameters**.
4. View the list of parameters by scrolling down.

Name	Value	Comment	Modified	Basic	Type	Category
blank_trimming	false				Bool...	Ansi Compliance
control_file_record_keep_time	7		✓		Integ...	Archiving and Recov...
db_create_online_log_dest_1			✓	✓	String	Archiving and Recov...
db_create_online_log_dest_2			✓	✓	String	Archiving and Recov...
db_create_online_log_dest_3			✓	✓	String	Archiving and Recov...
db_create_online_log_dest_4			✓	✓	String	Archiving and Recov...
db_create_online_log_dest_5			✓	✓	String	Archiving and Recov...
db_recovery_file_dest	/u01/app/oracle/fast_reco...		✓	✓	String	Archiving and Recov...
db_recovery_file_dest_size	12696M		✓	✓	Big I...	Archiving and Recov...

5. In the filter box, enter **job** (in this interface, capitalization doesn't matter) to filter the parameters list to show only those parameters that contain the word job. The `JOB_QUEUE_PROCESSES` initialization parameter should be listed.

Name	Container Name	Value	Modified	Dynamic	Session	PDB	Basic	Type	Description
max_datapump_jobs_per_pdb	*	100	<input checked="" type="checkbox"/>	Integer	maximum ...				
job_queue_processes	*	15	<input checked="" type="checkbox"/>	Integer	maximum ...				

6. Select the `JOB_QUEUE_PROCESSES` initialization parameter and click **Set**.
 7. In the Set Initialization Parameter dialog box, in the Value box, enter **4000**.



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

```
alter system set "job_queue_processes"=4000 CONTAINER=ALL scope=both sid='*' ;
```

- Question: What does the clause `SCOPE=BOTH` mean?
 Answer: The parameter value update is persistent now in memory and across the instance shutdown as the value is written to the SPFILE.
- Question: Is the parameter's value change applied to PDBs or only to the CDB?
 Answer: The parameter's updates are applied for the instance, hence for all containers of the CDB instance. Nevertheless, some parameters may hold values specific for some PDBs.
- Question: Another example: If the `DDL_LOCK_TIMEOUT` parameter is set to 10 in PDB1, would the value be persistent after the PDB is restarted?
 Answer: Yes, this is an example of another parameter whose value would be persistent in the PDB if you changed its value in the PDB. The PDB parameter values are stored in the `SYS.PDB_SPFILE$` table.

9. In the Set Initialization Parameter dialog box, click **OK**.

- Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.
10. In the Confirmation dialog box, verify that the message states "Parameter job_queue_processes successfully updated," and click **OK**.
 11. Notice that the `JOB_QUEUE_PROCESSES` value is updated to 4000.

The screenshot shows the 'Initialization Parameters' page in Oracle EM Express. The 'Current' tab is selected. A message at the top says, 'The parameter values listed here are currently used by open container(s).' Below are buttons for 'View', 'Set...', 'Validate with SPA', and 'Help'. A table lists parameters under sections 'Pluggable Database' and 'Scheduler'. The 'job_queue_processes' parameter is highlighted.

Name	Container Name	Value
max_datapump_jobs_per...	*	100
job_queue_processes	*	4000

12. In the upper-right corner, click **Log Out** to log out of EM Express.
13. Close the browser window.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 2-5 Shutting Down and Starting Up the Oracle Database

Overview

This practice lets you look more closely at shutting down and starting up your Oracle database and at what happens to the PDBs.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Complete the following steps on VM1.

1. Open a terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

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

```
$ sqlplus / as sysdba

...
SQL>
```

3. Show the current user. SQL*Plus is still running and the current user is `SYS`. If you were connected as `SYSTEM`, you would not be able to shut down the database instance.

```
SQL> SHOW USER

USER is "SYS"
SQL>
```

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

```
SQL> SHUTDOWN

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

5. Show the current user. 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 not available anymore.

```
SQL> SHOW con_name

ERROR:
ORA-01034: ORACLE not available
Process ID: 0
Session ID: 0 Serial number: 0

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

7. Start up the database instance in NOMOUNT mode.

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

```
SQL> STARTUP NOMOUNT

ORACLE instance started.

Total System Global Area 2466250752 bytes
Fixed Size                  8795760 bytes
Variable Size                671081088 bytes
Database Buffers            1778384896 bytes
Redo Buffers                 7979008 bytes
SQL>
```

8. Mount the database by using the ALTER DATABASE MOUNT command.

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

```
SQL> ALTER DATABASE MOUNT;

Database altered.
SQL>
```

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

```
SQL> ALTER DATABASE OPEN;

Database altered.
SQL>
```

10. Show the current container name.

```
SQL> SHOW con_name

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

11. Show the current user.

```
SQL> SHOW user

USER is "SYS"
SQL>
```

12. Check whether PDB1 is started up by querying the OPEN_MODE column in the V\$PDBS view. Notice that PDB1 is in MOUNTED mode, which means it is closed.

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

CON_ID NAME      OPEN_MODE
-----
2  PDB$SEED    READ ONLY
3  PDB1        MOUNTED
SQL>
```

13. List the data files for the entire CDB by querying the CDB_DATA_FILES view.

```
SQL> COLUMN file_name FORMAT A50
SQL> COLUMN tablespace_name FORMAT A10
SQL> SELECT file_name, tablespace_name FROM cdb_data_files;

FILE_NAME                                TABLESPACE
-----
/u01/app/oracle/oradata/ORCL/users01.dbf    USERS
/u01/app/oracle/oradata/ORCL/undotbs01.dbf   UNDOTBS1
/u01/app/oracle/oradata/ORCL/system01.dbf    SYSTEM
/u01/app/oracle/oradata/ORCL/sysaux01.dbf    SYSAUX
SQL>
```

14. Question: Why are the data files for PDB1 not listed in the previous step?
Answer: The data files for PDB1 are not listed because PDB1 is not opened.
15. Open PDB1. You can do this from within the root container by using the ALTER PLUGGABLE DATABASE command to change its open mode to OPEN. Alternatively, you can connect to PDB1 and open it from within itself.

```
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN;

Pluggable database altered.
SQL>
```

16. List the data files for PDB1.

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

FILE_NAME                                TABLESPACE
-----
/u01/app/oracle/oradata/ORCL/users01.dbf    USERS
/u01/app/oracle/oradata/ORCL/undotbs01.dbf   UNDOTBS1
/u01/app/oracle/oradata/ORCL/system01.dbf    SYSTEM
/u01/app/oracle/oradata/ORCL/sysaux01.dbf    SYSAUX
/u01/app/oracle/oradata/ORCL/PDB1/system01.dbf SYSTEM
/u01/app/oracle/oradata/ORCL/PDB1/sysaux01.dbf SYSAUX
/u01/app/oracle/oradata/ORCL/PDB1/undotbs01.dbf UNDOTBS1
/u01/app/oracle/oradata/ORCL/PDB1/users01.dbf  USERS

8 rows selected.
SQL>
```

17. It would be easier if PDB1 was automatically opened after instance startup. This would avoid the operation of opening it after each instance startup. Save PDB1's open state.
 - a. Check if PDB1 has a saved state by querying DBA_PDB_SAVED_STATES. The results show that PDB1 does not currently have a saved state.

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

no rows selected
SQL>
```

- b. Save the current state of PDB1.

```
SQL> ALTER PLUGGABLE DATABASE pdb1 SAVE STATE;

Pluggable database altered.
SQL>
```

- c. Verify that PDB1 now has a saved state by querying DBA_PDB_SAVED_STATES again. The query result indicates that the saved state is OPEN.

```
SQL> COL con_name FORMAT A12
SQL> COL instance_name FORMAT A12
SQL> COL state FORMAT A12
SQL> SELECT con_name, instance_name, state FROM dba_pdb_saved_states;

CON_NAME      INSTANCE_NAM STATE
-----        -----
PDB1          ORCL        OPEN
SQL>
```

18. Test that PDB1's open state persists after you restart the database instance.

- a. Shut down the database instance.

```
SQL> SHUTDOWN IMMEDIATE

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

- b. Start the database instance.

```
SQL> STARTUP

ORACLE instance started.
Total System Global Area 2466250752 bytes
Fixed Size                  8795760 bytes
Variable Size                671091088 bytes
Database Buffers            1778384896 bytes
Redo Buffers                 7979008 bytes
Database mounted.
Database opened.
SQL>
```

- c. Show the open modes for the PDBs in the CDB. PDB1 is in READ WRITE mode, which indicates that its saved state works.

```
SQL> SHOW PDBS
```

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

```
SQL>
```

- d. Question: How would you remove the saved state for PDB1?

Answer: The command `ALTER PLUGGABLE DATABASE pdb1 DISCARD STATE` reverts the state to the default behavior.

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

```
SQL> EXIT
```

```
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

Practice 2-6 Viewing Diagnostic Information

Overview

In this practice, you perform the following tasks:

- Examine the structure of the Automatic Diagnostic Repository (ADR)
- View the alert log two ways - first through a text editor and then using the Automatic Diagnostic Repository Command Interpreter (ADRCI)
- Configure the ADR space so that ADR diagnostics files do not exceed the disk space. The system administrator restricts the usage for ADR diagnostics files for the database, ORCL, to 200Mb. (This is a very small value used to demonstrate what happens in the case of the training session.)
- Enable DDL logging and log some DDL statements in the DDL log file

Tip:

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

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

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

Assumptions

You are logged in to VM1 as the oracle user.

Tasks

View the ADR Directories

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

In this section, you locate the XML and text-only versions of the alert log by querying the V\$DIAG_INFO view.

Complete the following tasks on VM1.

1. Open a terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL.

```
$ . oraenv  
ORACLE_SID = [oracle] ? ORCL
```

```
The Oracle base has been set to /u01/app/oracle  
$
```

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

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

3. View the locations of the various diagnostics directories in the ADR. The results below have been formatted for easier reading. The path that corresponds to the Diag Alert entry in the NAME column is for the XML version. This path is /u01/app/oracle/diag/rdbms/orcl/ORCL/alert. The path that corresponds to the Diag Trace entry is for the text-only version. This path is /u01/app/oracle/diag/rdbms/orcl/ORCL/trace. Notice that the path matches the default directory for the alert log, as stated in the tip at the beginning of this practice (\$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/trace).

```
SQL> SELECT name, value FROM v$diag_info;
NAME          VALUE
-----
Diag Enabled   TRUE
ADR Base       /u01/app/oracle
ADR Home       /u01/app/oracle/diag/rdbms/orcl/ORCL
Diag Trace     /u01/app/oracle/diag/rdbms/orcl/ORCL/trace
Diag Alert      /u01/app/oracle/diag/rdbms/orcl/ORCL/alert
Diag Incident   /u01/app/oracle/diag/rdbms/orcl/ORCL/incident
Diag Cdump      /u01/app/oracle/diag/rdbms/orcl/ORCL/cdump
Health Monitor  /u01/app/oracle/diag/rdbms/orcl/ORCL/hm
Default Trace File /u01/app/oracle/diag/rdbms/orcl/ORCL/trace/
ORCL_ora_24354.trc
Active Problem Count 1
Active Incident Count 1
11 rows selected.
SQL>
```

4. Exit SQL*Plus, but don't close the terminal window because you'll return to it later in the practice.

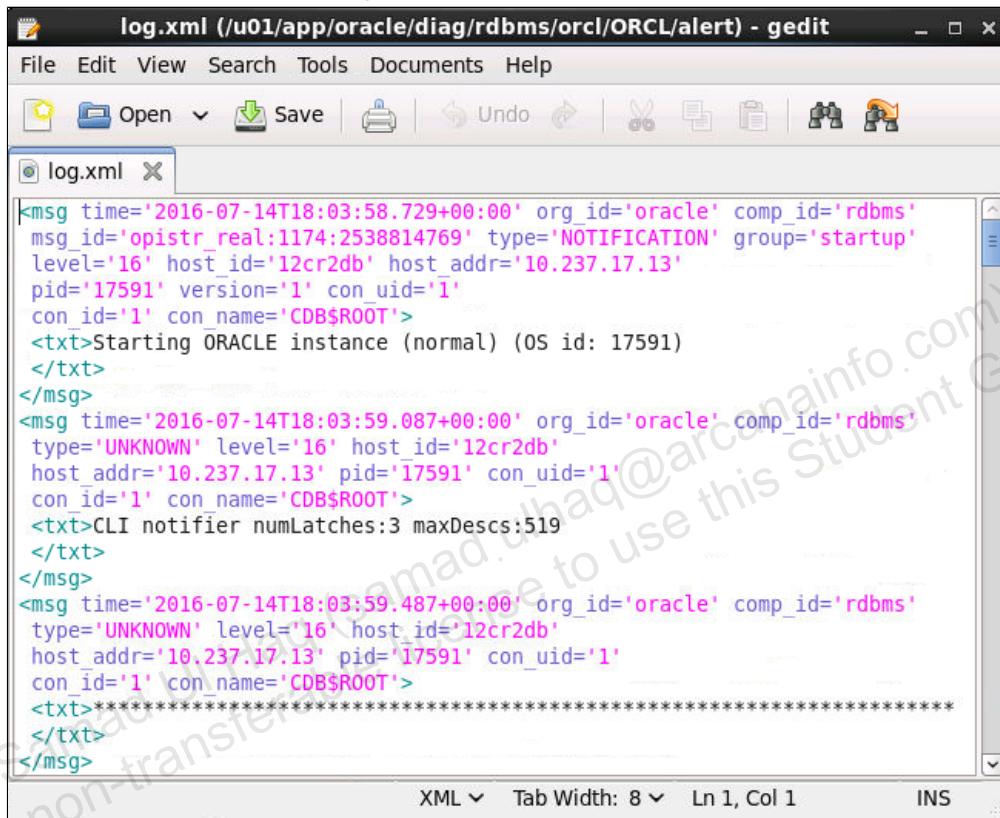
```
SQL> EXIT
...
$
```

View the Alert Log with Gedit

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

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

- a. On your desktop, open **Computer**, and then **Filesystem**. Browse to the `/u01/app/oracle/diag/rdbms/orcl/ORCL/alert` directory. Notice that there is a `log.xml` file in this directory.
- b. Right-click `log.xml`, and select **Open With**, and then **gedit**. Scroll through and examine 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.



The screenshot shows the gedit text editor window with the title bar "log.xml (/u01/app/oracle/diag/rdbms/orcl/ORCL/alert) - gedit". The menu bar includes File, Edit, View, Search, Tools, Documents, Help. The toolbar includes Open, Save, Undo, Redo, Cut, Copy, Paste, Find, Replace, and Find Next. The main text area displays the XML content of the log file:

```
<msg time='2016-07-14T18:03:58.729+00:00' org_id='oracle' comp_id='rdbms'
msg_id='opistr_real:1174:2538814769' type='NOTIFICATION' group='startup'
level='16' host_id='12cr2db' host_addr='10.237.17.13'
pid='17591' version='1' con_uid='1'
con_id='1' con_name='CDB$ROOT'>
<txt>Starting ORACLE instance (normal) (OS id: 17591)
</txt>
</msg>
<msg time='2016-07-14T18:03:59.087+00:00' org_id='oracle' comp_id='rdbms'
type='UNKNOWN' level='16' host_id='12cr2db'
host_addr='10.237.17.13' pid='17591' con_uid='1'
con_id='1' con_name='CDB$ROOT'>
<txt>CLI notifier numLatches:3 maxDescs:519
</txt>
</msg>
<msg time='2016-07-14T18:03:59.487+00:00' org_id='oracle' comp_id='rdbms'
type='UNKNOWN' level='16' host_id='12cr2db'
host_addr='10.237.17.13' pid='17591' con_uid='1'
con_id='1' con_name='CDB$ROOT'>
<txt>*****
</txt>
</msg>
```

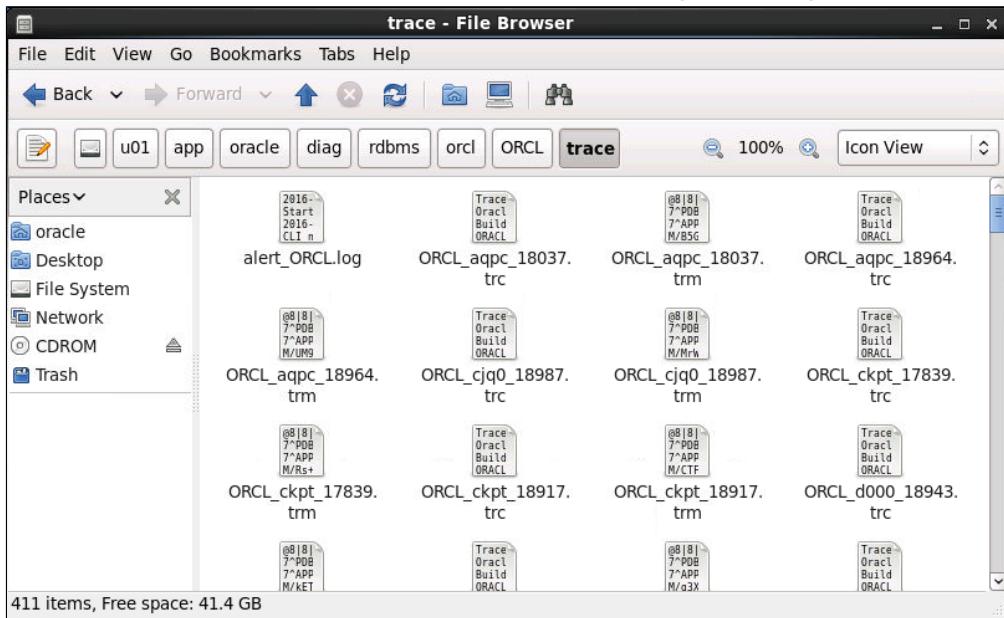
The status bar at the bottom shows "XML" and "INS".

- c. Select **File** and then **Quit** to close gedit.

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

- a. Browse to `/u01/app/oracle/diag/rdbms/orcl/ORCL/trace`. The `alert_ORCL.log` (format is `alert_SID.log`) file is the text-only version.
In this directory, you also have server process trace files (TRC files) and trace map files (TRM files). Each server and background process can write to an associated trace file. When a process detects an

internal error, it dumps information about the error to its trace file. Trace map files contain structural information about trace files and are used for searching and navigation.



- Right-click `alert_ORCL.log`, and open the file with gedit. Scroll down and view the contents of the alert log.

```

2016-07-14T18:03:58.729593+00:00
Starting ORACLE instance (normal) (OS id: 17591)
2016-07-14T18:03:59.087993+00:00
CLI notifier numLatches:3 maxDescs:519
2016-07-14T18:03:59.487991+00:00
*****
2016-07-14T18:03:59.488113+00:00
Dump of system resources acquired for SHARED GLOBAL AREA (SGA)

2016-07-14T18:03:59.488327+00:00
Per process system memlock (soft) limit = 64K
2016-07-14T18:03:59.488444+00:00
Expected per process system memlock (soft) limit to lock
SHARED GLOBAL AREA (SGA) into memory: 2354M
2016-07-14T18:03:59.488665+00:00
Available system pagesizes:
 4K, 2048K
2016-07-14T18:03:59.488908+00:00
Supported system pagesize(s):
2016-07-14T18:03:59.489024+00:00
  PAGESIZE  AVAILABLE_PAGES  EXPECTED_PAGES  ALLOCATED_PAGES  ERROR(s)
2016-07-14T18:03:59.489140+00:00

```

- Find information about the ORCL instance in the alert log. First, find out when the instance had last started and the startup modes it went through. Next, view the non-default system parameter values that the instance used and the background processes that started.

- Using the search feature in gedit (binoculars button on the toolbar), search for the text **Starting ORACLE instance**. Click the **Find** button repeatedly to locate each occurrence. If you click the very end

of the file and then use the *Search backwards* feature with the same search text in the Find dialog box, you should be able to locate the line that tells you when the instance last started. Your results will be different than those shown below.

The screenshot shows a terminal window titled 'alert_ORCL.log' displaying Oracle database log messages. A 'Find' dialog box is overlaid on the window, with the search term 'Starting ORACLE instance' entered. The 'Search backwards' checkbox is checked. The log messages include:

```

ARCH: Archival disabled due to shutdown: 1089
Shutting down archive processes
Archiving is disabled
2016-07-14T18:10:24.033889+00:00
JIT: pid 18779 requesting stop
2016-07-14T18:10:25.034104+00:00
Stopping background process VKTM
2016-07-14T18:10:25.058748+00:00
ARCH: Archival disabled due to shutdown: 1089
Shutting down archive processes
Archiving is disabled
JIT: pid 18779 requesting stop
2016-07-14T18:10:32.832236+00:00
Instance shutdown complete (OS id: 18779)
2016-07-14T18:10:37.112278+00:00
Starting ORACLE instance (normal) (OS id: 18873)
2016-07-14T18:10:37.115511+00:00
CLI notifier numLatches:3 maxDescs:519
2016-07-14T18:10:37.119094+00:00
*****
2016-07-14T18:10:37.119250+00:00
Dump of system resources acquired for SHARED GLOBAL AREA (SGA)

2016-07-14T18:10:37.119457+00:00
Per process system memlock (soft) limit = 64K
2016-07-14T18:10:37.119577+00:00
Expected per process system memlock (soft) limit to lock
SHARED GLOBAL AREA (SGA) into memory: 2354M
2016-07-14T18:10:37.119801+00:00
Available system pagesizes:

```

Plain Text ▾ Tab Width: 8 ▾ Ln 1520, Col 1 INS

- To view information about the startup modes, enter **ALTER DATABASE** in the "Search for" text box, uncheck **Search backwards**, and click **Find**. The search finds **ALTER DATABASE MOUNT** and then a few lines later, **ALTER DATABASE OPEN**. Therefore, the startup modes are **MOUNT** and **OPEN**. Close the Find dialog box.

The screenshot shows a terminal window titled 'alert_ORCL.log' displaying Oracle database log messages. A 'Find' dialog box is overlaid on the window, with the search term 'ALTER DATABASE' entered. The 'Search backwards' checkbox is unchecked. The log messages include:

```

MMON started with pid=32, OS id=18939
2016-07-14T18:10:40.050538+00:00
starting up 1 dispatcher(s) for network address '(ADDRESS=(PARTIAL=YES)(PROTOCOL=TCP))'...
2016-07-14T18:10:40.052338+00:00
MMNL started with pid=31, OS id=18941
starting up 1 shared server(s) ...
Starting background process TMON
ORACLE_BASE from environment = /u01/app/oracle
2016-07-14T18:10:40.148807+00:00
TMON started with pid=35, OS id=18947
2016-07-14T18:10:40.325998+00:00
ALTER DATABASE MOUNT
2016-07-14T18:10:42.460501+00:00
Using default pga aggregate limit of 2048 MB
2016-07-14T18:10:44.399157+00:00
Successful mount of redo thread 1, with mount id 1444960672
2016-07-14T18:10:44.399764+00:00
Database mounted in Exclusive Mode
Lost write protection disabled
Using STANDBY ARCHIVE_DEST parameter default value as USE_DB_RECOVERY_FILE_DEST
Completed: ALTER DATABASE MOUNT
2016-07-14T18:10:44.530567+00:00
ALTER DATABASE OPEN
2016-07-14T18:10:44.537703+00:00
Ping without log force is disabled:
  instance mounted in exclusive mode.
Buffer Cache Full DB Caching mode changing from FULL CACHING DISABLED to FULL CACHING ENABLED
Endian type of dictionary set to little
2016-07-14T18:10:44.583933+00:00
Thread 1 opened at log sequence 2

```

- c. Scroll up to view the list of background processes that started.

```

alert_ORCL.log X
=====
NOTE: PatchLevel of this instance 0
=====
Starting background process PMON
2016-07-14T18:10:38.074612+00:00
PMON started with pid=2, OS id=18876
Starting background process CLMN
2016-07-14T18:10:38.104907+00:00
CLMN started with pid=3, OS id=18878
Starting background process PSP0
2016-07-14T18:10:38.134552+00:00
Starting background process VKTM
2016-07-14T18:10:38.134582+00:00
PSP0 started with pid=4, OS id=18880
2016-07-14T18:10:39.237254+00:00
VKTM started with pid=5, OS id=18883 at elevated (RT) priority
2016-07-14T18:10:39.237299+00:00
Starting background process GEN0
2016-07-14T18:10:39.240215+00:00
VKTM running at (1)millisec precision with DBRM quantum (100)ms
Starting background process MMAN
2016-07-14T18:10:39.264263+00:00
GEN0 started with pid=6, OS id=18887
2016-07-14T18:10:39.289028+00:00
MMAN started with pid=7, OS id=18889
Starting background process GEN1
Starting background process DIAG
2016-07-14T18:10:39.354899+00:00
GEN1 started with pid=9, OS id=18893_18895

```

- d. Scroll up a bit more and view the system parameters with non-default values.

```

alert_ORCL.log X
=====
Using parameter settings in server-side spfile /u01/app/oracle/product/12.2.0/dbhome_1/dbs/spfileORCL.ora
System parameters with non-default values:
processes          = 300
nls_language       = "AMERICAN"
nls_territory      = "AMERICA"
sga_target         = 2352M
control_files      = "/u01/app/oracle/oradata/ORCL/control01.ctl"
control_files      = "/u01/app/oracle/fast_recovery_area/ORCL/control02.ctl"
db_block_size      = 8192
compatible         = "12.2.0"
db_recovery_file_dest = "/u01/app/oracle/fast_recovery_area/ORCL"
db_recovery_file_dest_size= 13176M
undo_tablespace    = "UNDOTBS1"
remote_login_passwordfile= "EXCLUSIVE"
db_domain          = "example.com"
instance_name       = "ORCL"
dispatchers        = "(PROTOCOL=TCP) (SERVICE=ORCLXDB)"
local_listener     = "LISTENER_ORCL"
audit_file_dest    = "/u01/app/oracle/admin/ORCL/adump"
audit_trail         = "DB"
db_name             = "ORCL"
open_cursors        = 300
pga_aggregate_target = 783M
diagnostic_dest    = "/u01/app/oracle"
enable_pluggable_database= TRUE
NOTE: remote asm mode is local (mode 0x1; from cluster type)
2016-07-14T18:10:37.454822+00:00

```

- e. Select **File**, and then **Quit** to close gedit.
f. Close the trace - File Browser window.

[View the Alert Log with ADRCI](#)

1. Return to the terminal window and start the ADRCI tool. We'll refer to this terminal as Terminal 1. Recall that you've already set the Oracle environment variables at the beginning of this practice; however, only the ORACLE_HOME environment variable needs to be set prior to starting ADRCI. If you ever need to set just

that one variable, you can do so by entering the following at the command prompt: `export PATH=$PATH:$ORACLE_HOME/bin`.

```
$ adrci
```

```
ADRCI: Release 12.2.0.1.0 - Production on Mon Jul 25 21:12:34 2016
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.
ADR base = "/u01/app/oracle"
adrci>
```

- View the alert log by using the `SHOW ALERT` command. When prompted, select option **2** to chose the `diag/rdbms/orcl/ORCL` directory. The `SHOW ALERT` command opens the alert log file in the vi editor, by default.

```
adrci> SHOW ALERT
Choose the home from which to view the alert log:
1: diag/tnslsnr/12cr2db/listener
2: diag/rdbms/orcl/ORCL
3: diag/clients/user_oracle/host_1896032729_105
Q: to quit
Please select option: 2
Output the results to file: /tmp/alert_2464_13994_ORCL_1.ado
2016-04-21 16:11:55.864000 +00:00
Starting ORACLE instance (normal) (OS id: 9655)
CLI notifier numLatches:3 maxDescs:519
*****
Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
Per process system memlock (soft) limit = 64K
Expected per process system memlock (soft) limit to lock
SHARED GLOBAL AREA (SGA) into memory: 2354M
Available system pagesizes:
 4K, 2048K
Supported system pagesize(s):
  PAGE_SIZE  AVAILABLE_PAGES  EXPECTED_PAGES  ALLOCATED_PAGES  ERROR(s)
    4K        Configured          3            602115        NONE
   2048K                0            1177            0        NONE
RECOMMENDATION:
1. For optimal performance, configure system with expected number of pages for
every supported system pagesize prior to the next instance restart operation.
2. Increase per process memlock (soft) limit to at least 2354MB to lock 100% of
SHARED GLOBAL AREA (SGA) pages into physical memory
*****
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
"/tmp/alert_2464_13994_ORCL_1.ado" 3575L, 199325C
```

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

```
...
ORACLE_BASE from environment = /u01/app/oracle
ALTER DATABASE MOUNT
2016-05-05 18:31:45.155000 +00:00
Using default pga_aggregate_limit of 2048 MB
```

```
2016-05-05 18:31:47.004000 +00:00
Successful mount of redo thread 1, with mount id 1438853390
Database mounted in Exclusive Mode
Lost write protection disabled
Completed: ALTER DATABASE MOUNT
ALTER DATABASE OPEN
Ping without log force is disabled:
  instance mounted in exclusive mode.
Endian type of dictionary set to little
Starting background process TMON
TMON started with pid=33, OS id=1673
Thread 1 opened at log sequence 123
  Current log# 3 seq# 123 mem# 0: /u01/app/oracle/oradata/ORCL/redo03.log
Successful open of redo thread 1
MTTR advisory is disabled because FAST_START_MTTR_TARGET is not set
TT00: Gap Manager starting (PID:1675)
[1671] Successfully onlined Undo Tablespace 2.
Undo initialization finished serial:0 start:1232765499 end:1232765576 diff:77 ms
(0.1 seconds)
Verifying minimum file header compatibility for tablespace encryption for pdb
1..
Verifying file header compatibility for tablespace encryption completed for pdb
1
Database Characterset is AL32UTF8
No Resource Manager plan active
2016-05-05 18:31:48.302000 +00:00
replication_dependency_tracking turned off (no async multimaster replication
found)
Starting background process AQPC
AQPC started with pid=37, OS id=1681
Endian type of dictionary set to little
Database Characterset for PDB$SEED is AL32UTF8
2016-05-05 18:31:49.434000 +00:00
Opatch validation is skipped for PDB PDB$SEED (con_id=0)
Opening pdb with no Resource Manager plan active
Cannot start service pdb1.example.com, reason=13
Starting background process CJQ0
Completed: ALTER DATABASE OPEN
CJQ0 started with pid=38, OS id=1701
2016-05-05 18:32:12.856000 +00:00
Shared IO Pool defaulting to 128MB. Trying to get it from Buffer Cache for
process 1652.
=====
Dumping current patch information
=====
No patches have been applied
=====
db_recovery_file_dest_size of 12696 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.

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

```
...
Starting ORACLE instance (normal) (OS id: 1548)
CLI notifier numLatches:3 maxDescs:519
*****
Dump of system resources acquired for SHARED GLOBAL AREA (SGA)
Per process system memlock (soft) limit = 64K
Expected per process system memlock (soft) limit to lock
SHARED GLOBAL AREA (SGA) into memory: 2354M
Available system pagesizes:
 4K, 2048K
Supported system pagesize(s):
  PAGE_SIZE  AVAILABLE_PAGES  EXPECTED_PAGES  ALLOCATED_PAGES  ERROR(s)
    4K          Configured            3             602115        NONE
  2048K                0              1177             0        NONE
RECOMMENDATION:
 1. For optimal performance, configure system with expected number
 of pages for every supported system pagesize prior to the next
 instance restart operation.
 2. Increase per process memlock (soft) limit to at least 2354MB
 to lock 100% of SHARED GLOBAL AREA (SGA) pages into physical memory
*****
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
```

6. 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
2016-05-05 18:31:45.155000 +00:00
Using default pga_aggregate_limit of 2048 MB
2016-05-05 18:31:47.004000 +00:00
Successful mount of redo thread 1, with mount id 1438853390
Database mounted in Exclusive Mode
Lost write protection disabled
Completed: ALTER DATABASE MOUNT
```

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

```

ALTER DATABASE OPEN
Ping without log force is disabled:
  instance mounted in exclusive mode.
Endian type of dictionary set to little
Starting background process TMON
Endian type of dictionary set to little
Starting background process TMON
TMON started with pid=33, OS id=1673
Thread 1 opened at log sequence 123
  Current log# 3 seq# 123 mem# 0: /u01/app/oracle/oradata/ORCL redo03.log
Successful open of redo thread 1
MTTR advisory is disabled because FAST_START_MTTR_TARGET is not set
TT00: Gap Manager starting (PID:1675)
[1671] Successfully onlined Undo Tablespace 2.
Undo initialization finished serial:0 start:1232765499 end:1232765576 diff:77 ms
(0.1 seconds)
Verifying minimum file header compatibility for tablespace encryption for pdb
1..
Verifying file header compatibility for tablespace encryption completed for pdb
1
Database Charcterset is AL32UTF8
No Resource Manager plan active
2016-05-05 18:31:48.302000 +00:00
replication_dependency_tracking turned off (no async multimaster replication
found)
Starting background process AQPC
AQPC started with pid=37, OS id=1681
Endian type of dictionary set to little
Database Charcterset for PDB$SEED is AL32UTF8
2016-05-05 18:31:49.434000 +00:00
Opatch validation is skipped for PDB PDB$SEED (con_id=0)
Opening pdb with no Resource Manager plan active
Cannot start service pdb1.example.com, reason=13
Starting background process CJQ0
Completed: ALTER DATABASE OPEN
CJQ0 started with pid=38, OS id=1701
2016-05-05 18:32:12.856000 +00:00
Shared IO Pool defaulting to 128MB. Trying to get it from Buffer Cache for
process 1652.

```

8. Exit the vi editor by entering :q and pressing **Enter**.
9. Exit ADRCI by entering **Q**, and then **exit**.

```

Choose the home from which to view the alert log:
1: diag/tnslsnr/12cr2db/listener1
2: diag/rdbms/orcl/ORCL
3: diag/clients/user_oracle/host_1896032729_105
Q: to quit
Please select option: Q

```

adrci>

Limit the Target Size for ADR Diagnostic Files

1. Find the retention policy values.

```
adrci> SET HOMEPATH diag/rdbms/orcl/ORCL
adrci> SELECT sizep_policy FROM adr_control_aux;

ADR Home = /u01/app/oracle/diag/rdbms/orcl/ORCL:
*****
SIZEP_POLICY
-----
18446744073709551615
1 row fetched

adrci> SELECT shortp_policy, longp_policy FROM adr_control;

ADR Home = /u01/app/oracle/diag/rdbms/orcl/ORCL:
*****
SHORTP_POLICY      LONGP_POLICY
-----          -----
720                8760
1 row fetched

adrci>
```

2. Limit the target size for ADR ORCL diagnostics files to 200MB.

```
adrci> SET CONTROL (SIZEP_POLICY = 200000000)
adrci> SELECT sizep_policy FROM adr_control_aux;

ADR Home = /u01/app/oracle/diag/rdbms/orcl/ORCL:
*****
SIZEP_POLICY
-----
200000000
1 row fetched

adrci>
```

Observe that the values for SHORTP_POLICY and LONGP_POLICY are in hours, which are the default values. The hour values translate to 30 days for SHORTP_POLICY and 365 days for LONGP_POLICY.

3. Predict how much space is going to be required to store the diagnostics for ORCL for a short retention period set to 8 days. Your value for Size Policy Bytes may differ than the one below.

```
adrci> ESTIMATE (SHORTP_POLICY = 192)
Estimate
Short Policy Hours: 192
Long Policy Hours: 8760
Size Policy Bytes: 21311581
adrci>
```

The ADR advisor says that for a short retention policy set to 8 days and a long retention policy kept to 365 days, only 21 MB would be required. This prediction relies on the current ADR diagnostics files space used.

4. What would the advisor predict for a long retention period set to 90 days? Your value for Size Policy Bytes may differ than the one below.

```
adrci> ESTIMATE (SHORTP_POLICY = 192, LONGP_POLICY = 2160)
Estimate
Short Policy Hours: 192
Long Policy Hours: 2160
Size Policy Bytes: 19635483
adrci>
```

The ADR advisor says that for a short retention policy set to 8 days and a long retention policy kept to 90 days, only 19 MB would be required. This prediction relies on the current ADR diagnostics files space used.

5. Which short and long retention periods would the advisor predict for a ADR size target of 200MB? Your hour values may differ than the results below.

```
adrci> ESTIMATE (SIZEP_POLICY = 209715200)
Estimate
Short Policy Hours: 2346
Long Policy Hours: 28543
Size Policy Bytes: 209715200
adrci>
```

The ADR advisor says that for a size target set to 200MB, the possible long retention could be set to 28543 days and the short retention could be set to 2346 days.

6. Set the ADR target size to 100MB.

```
adrci> SET CONTROL (sizep_policy = 104857600)
adrci>
```

7. Purge the ADR files down to 5MB, without permanently changing the current retention policy settings.

- Open another terminal window (which we'll refer to as Terminal 2) and set the environment variables for your ORCL database.

```
$ . oraenv
ORACLE_SID = [ORCL] ? ORCL
```

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

- Check if in the ORCL ADR directory of your own database, the space used by ADR files exceeds 5MB. If this is not the case, find an appropriate value so that you can verify later that sufficient ADR files were removed from the ADR to reach the target. The results below indicate that the current space used by ADR files exceeds 5MB. In fact, the space exceeds 60MB. Your values might differ from the example values below.

```
$ cd $ORACLE_BASE/diag/rdbms/orcl/ORCL  
$ du -bs .  
64478694 .  
$ du -hs .  
66M .  
$
```

- c. In Terminal 1 (your adrci session), purge the ADR log files down to 5MB.

```
adrci> PURGE -size 5000000 -type log  
adrci>
```

- d. In Terminal 2, check that the command removed the ADR files to release space in the ADR. The results indicate that the command did not remove the ADR files. The space used by ADR is still the same. The du -bs value might differ slightly from the value in Step 7b, and they both might differ from the example values below.

```
$ du -bs .  
64478694 .  
$ du -hs .  
66M .  
$
```

- e. In Terminal 2, get help with the PURGE command. Notice that when purging by size, only INCIDENT, TRACE, CDUMP, and UTSCDMP data is considered.

```
adrci> HELP PURGE
```

```
Usage: PURGE [[-i <id1> | <id1> <id2>] |
               [-age <mins>] |
               [-size <bytes>] |
               [-type {ALERT|INCIDENT|TRACE|CDUMP|HM|UTSCDMP|LOG} ]]
```

Purpose: Purge the diagnostic data in the current ADR home. If no option is specified, the default purging policy will be used.

Options:

[-i id1 | id1 id2]: Purge a single specified incident, or a range of incidents.

[-age <mins>]: Purge diagnostic data older than <mins> from the ADR home, if the data is purgable.

[-size <bytes>]: Purge diagnostic data from the ADR home until the size of the home reaches <bytes> bytes.

[-type ALERT|INCIDENT|TRACE|CDUMP|HM|UTSCDMP|LOG]: Purge a specific type of data.

Notes:

When purging by size, only INCIDENT, TRACE, CDUMP and UTSCDMP data is considered.

Some data cannot be purged (such as incidents in the 'tracked' state), which means that the specified target size may not be reached in all cases.

Examples:

```
purge
purge -i 123 456
purge -age 60 -type incident
purge -size 10000000
```

```
adrci>
```

- f. In Terminal 1, try the PURGE command again, but this time, leave off the -type log part at the end.

```
adrci> PURGE -size 5000000
adrci>
```

- g. In Terminal 2, view the space used by ADR. Your result may be slightly different than the result shown below.

```
$ du -hs .
5.7M .
$
```

- h. Question: Did the the PURGE command achieved what was expected?

Answer: It removed ADR files but could not reach the target size requested. Some ADR files cannot be removed. ADR contains a lot of different files: logs, traces, dumps, reports, metadata, internal repository files (such as lock files and relation files). Those files are purged in different ways. Some are always present, and cannot be purged at all. Others are purged by simply deleting the entire file. Some files, such as relation files, are "purged" by purging parts of their internal structure.

- i. In Terminal 1, exit adrci and close the terminal window.

```
adrci > exit
$
```

Log DDL statements in the DDL Log File

1. Determine if DDL logging is enabled in PDB1. If not, enable it by setting the value for the `ENABLE_DDL_LOGGING` initialization parameter to TRUE.
 - a. In Terminal 2, start SQL*Plus and log in to the database as the `SYS` user with the `SYSDBA` privilege.


```
$ sqlplus / as sysdba
...
SQL>
```
 - b. Switch to PDB1.


```
SQL> ALTER SESSION SET CONTAINER = PDB1;
Session altered.
SQL>
```
 - c. Issue the `SHOW PARAMETER` command to view the value for `ENABLE_DDL_LOGGING`. The results show that this parameter is not enabled.


```
SQL> SHOW PARAMETER enable_ddl_logging
NAME          TYPE    VALUE
-----        -----  -----
enable_ddl_logging boolean FALSE
SQL>
```
 - d. Enable DDL logging for just this session by using the `ALTER SESSION` command.


```
SQL> ALTER SESSION SET enable_ddl_logging = TRUE;
Session altered.
SQL>
```
 - e. Issue the `SHOW PARAMETER` command again and verify that the value for `ENABLE_DDL_LOGGING` is TRUE.


```
SQL> SHOW PARAMETER enable_ddl_logging
NAME          TYPE    VALUE
-----        -----  -----
enable_ddl_logging boolean TRUE
SQL>
```
2. Create and drop a table to generate DDL statements.

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

SQL> DROP TABLE TEST;

Table dropped.

SQL>
```

3. Exit SQL*Plus.

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

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

```
$ cd /u01/app/oracle/diag/rdbms/orcl/ORCL/log
```

5. List the contents of the log directory.

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

6. View the `ddl_ORCL.log` file by using the CAT command. Your data will be different than the data shown below.

```
$ cat ddl_ORCL.log
2016-11-29T22:53:33.942950+00:00
diag_adl:CREATE TABLE TEST (name varchar2(15) )
2016-11-29T22:53:56.421792+00:00
diag_adl:DROP TABLE TEST
$
```

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

```
$ cd ddl
$ ls
log.xml
$
```

8. Close all the terminal windows.

3

Creating PDBs



ORACLE®

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Objectives for Lesson 3

After completing this lesson, you should be able to:

- Describe the methods and tools used to create PDBs
- Create PDBs from seed with SQL*Plus
- Clone PDBs with SQL*Plus
- Unplug and plug in PDBs with SQL*Plus
- Drop PDBs with SQL*Plus



ORACLE®

Methods and Tools to Create PDBs

- Methods to create PDBs:
 - Create a PDB by using the seed
 - Create a PDB from a non-CDB
 - Clone an existing PDB or non-CDB
 - Plug an unplugged PDB into a different CDB
 - Relocate a PDB to a different CDB
 - Create a PDB as a proxy PDB
- Tools to create PDBs:
 - SQL*Plus
 - SQL Developer
 - Enterprise Manager Cloud Control
 - DBCA - Create a PDB from seed or by using the unplug/plug method.

ORACLE®

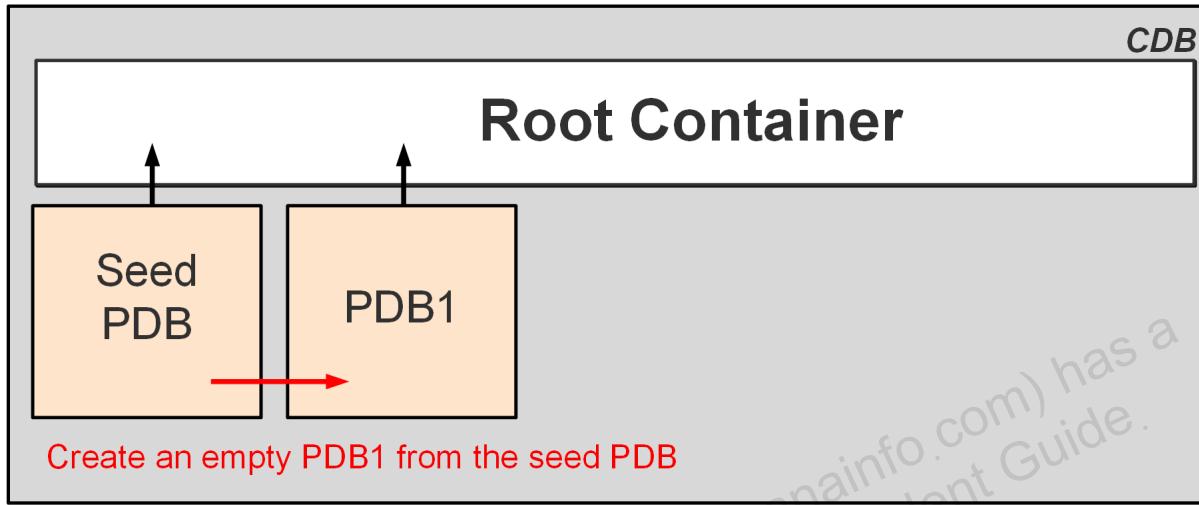
You can use the following methods to create a PDB:

- **Create a PDB by using the seed:** Create a PDB in a CDB using the files of the CDB seed or application seed. This technique copies the files associated with the seed to a new location and associates the copied files with the new PDB.
- **Create a PDB from a non-CDB:** Create a PDB by adopting a non-CDB into a PDB. You can use the DBMS_PDB package to create an unplugged PDB from an Oracle Database 12c non-CDB. You can then plug the unplugged PDB into the CDB.
- **Clone an existing PDB or non-CDB:** Create a PDB by cloning a source PDB or non-CDB. A source can be a PDB in the local CDB, a PDB in a remote CDB, a PDB in a local or remote application container, or a non-CDB. This technique copies the files associated with the source to a new location and associates the copied files with the new PDB.
- **Plug an unplugged PDB into a CDB:** Create a PDB by using the XML metadata file that describes the PDB and the files associated with the PDB to plug it into the CDB.
- **Relocate a PDB to a different CDB:** Create a PDB by relocating it from one CDB to another. This technique moves the files associated with the PDB to a new location.
- **Create a PDB as a proxy PDB:** Create a PDB as a proxy PDB by referencing a different PDB with a database link. The referenced PDB can be in the same CDB as the proxy PDB, or it can be in a different CDB.

Note: This course focuses on how to use SQL*Plus to create a PDB from seed and clone a PDB.

Creating PDBs from Seed

- You can create a new empty PDB by using the seed PDB as a template.
 - Every CDB has a seed PDB.
- To create a PDB from seed with SQL*Plus, use the `CREATE PLUGGABLE DATABASE` statement.



ORACLE®

To create a PDB from seed with SQL*Plus, you use the `CREATE PLUGGABLE DATABASE` statement. The creation of a new PDB from the seed is nearly instantaneous. This action copies the data files from the `READ ONLY` seed PDB to the target directory defined in the `CREATE PLUGGABLE DATABASE` statement. It creates tablespaces such as `SYSTEM`, to store a full catalog including metadata pointing to Oracle-supplied objects, and `SYSAUX` for local auxiliary data. It creates default schemas and common users that exist in seed PDB, `SYS` who continues to have all superuser privileges and `SYSTEM` who can administer the PDB. A new default service is created for the PDB.

Prerequisites for using the `CREATE PLUGGABLE DATABASE` statement include:

- You must be connected to a CDB and the current container must be the root.
- You must have the `CREATE PLUGGABLE DATABASE` system privilege.
- The CDB in which the PDB is being created must be in `READ WRITE` mode.

View:

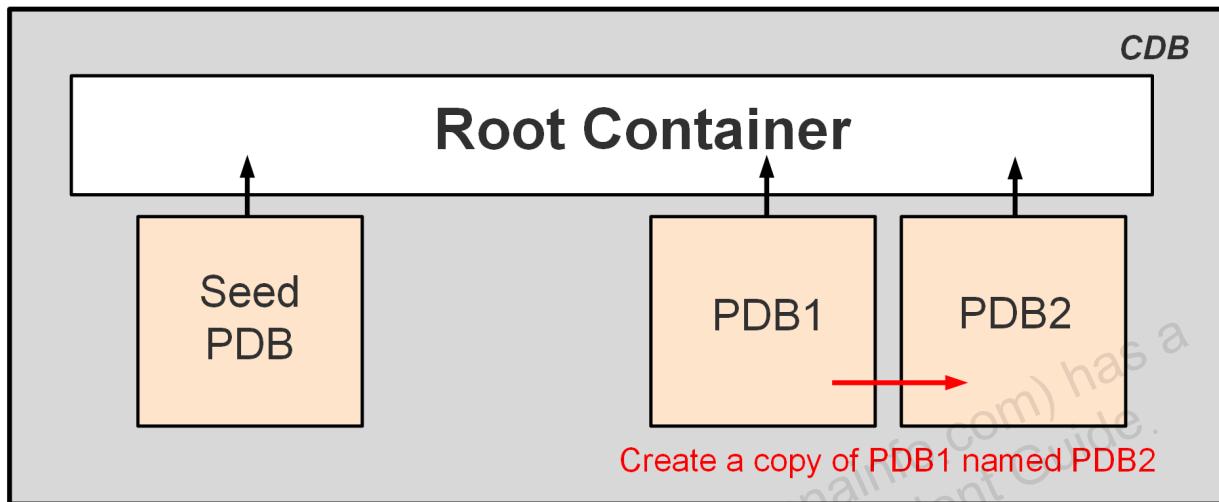
View the following example in *Oracle Database SQL Language Reference* to learn how to create a PDB by using the seed PDB as a template. You can create a PDB from seed in many ways. This example shows you how to include a default tablespace, which is not included in the seed PDB. It also shows you how to use the FILE_NAME_CONVERT clause to generate similar names for and paths to data files as in the seed PDB; for example, system01.dbf, sysaux01.dbf, and undotbs01.dbf.

- `CREATE PLUGGABLE DATABASE` (Scroll down to "Creating a PDB by Using the Seed: Example")

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Cloning PDBs

- Cloning is copying a source PDB from a CDB and plugging the copy into the same CDB or another CDB.
- Example: PDB1 is cloned as PDB2 in the same CDB. The seed PDB, while present in the CDB, is not used.



ORACLE

Cloning is suitable for the following situations:

- You want to test the application patch of your production PDB. You first clone your production application in a cloned PDB, patch the cloned PDB, and test it.
- You want to diagnose performance issues or perform performance regression tests on your application. Because you cannot perform this operation in parallel with the production in the same database, you clone the PDB into another CDB.

To clone a PDB with SQL*Plus, you use the `CREATE PLUGGABLE DATABASE` statement.

Prerequisites for using the `CREATE PLUGGABLE DATABASE` statement to clone a PDB include:

- You must be connected to a CDB and the current container must be the root.
- You must have the `CREATE PLUGGABLE DATABASE` system privilege.
- The CDB in which the PDB is being created must be in `READ WRITE` mode.
- You must put the PDB being cloned into `READ ONLY` mode before you can clone it.

 **View:**

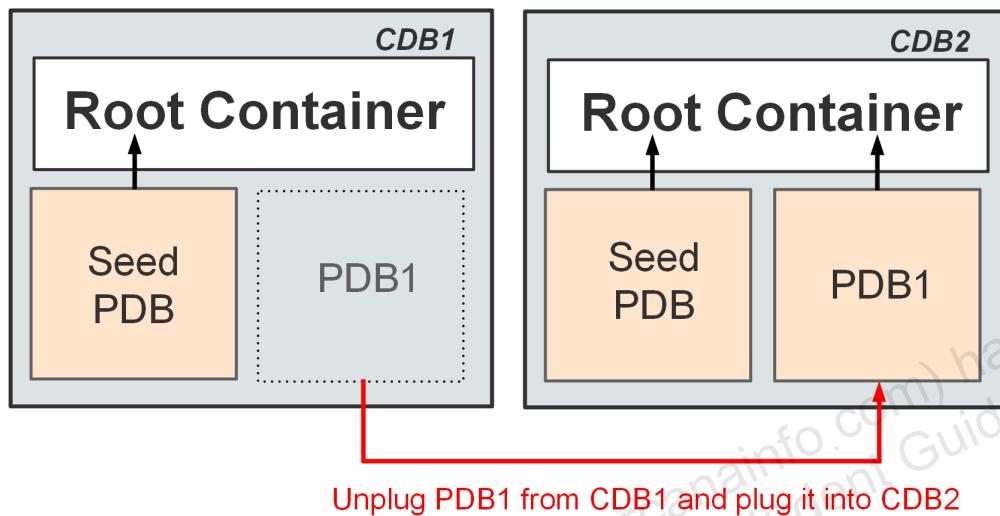
You have many options when cloning a PDB. Review the following example in *Oracle Database SQL Language Reference*. This example creates unlimited storage for the new PDB and also copies the exact data file names from the source PDB by using the FILE_NAME_CONVERT clause.

- [CREATE PLUGGABLE DATABASE](#) (Scroll down to Cloning a PDB From an Existing PDB: Example")

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Unplugging and Plugging In PDBs

- Unplugging a PDB is disassociating the PDB from its CDB.
- Plugging in a PDB is associating a PDB with a CDB.
- You can plug a PDB into the same or another CDB.
- Example: PDB1 is unplugged from CDB1 and plugged into CDB2.



ORACLE®

Unauthorized reproduction or distribution prohibited. Copyright © 2017, Oracle and/or its affiliates.

- You have to upgrade a PDB to the latest Oracle version, but you do not want to apply it on all PDBs. Instead of upgrading a CDB from one release to another, you can unplug a PDB from one Oracle Database release, and then plug it into a newly created CDB from a later release.
- You want to test the performance of the CDB without a particular PDB. You unplug the PDB, test the performance without the PDB and, if necessary, replug the PDB into the CDB.
- You want to maintain a collection of PDB “gold images” as unplugged PDBs.

You use the `ALTER PLUGGABLE DATABASE` statement to unplug a PDB from a CDB. To unplug a PDB, you must first close it, and then generate an XML manifest file. The XML file contains the names and full paths of the tablespaces and data files of the unplugged PDB. That information is then used by the plugging operation. You use the `CREATE PLUGGABLE DATABASE` statement to plug a PDB into a CDB.

View:

Review the following examples of unplugging and plugging a PDB in *Oracle Database SQL Language Reference*. Unplugging is fairly simple with minimal SQL required. With plugging, you have several options to consider.

- `ALTER PLUGGABLE DATABASE` (Scroll down to "Unplugging a PDB from a CDB: Example")
- `CREATE PLUGGABLE DATABASE` (Scroll down to "Plugging a PDB into a CDB: Example")

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Dropping PDBs

- When you drop a PDB, you remove all references to it and its data files in the control file of the CDB.
- Archived logs and backups associated with the dropped PDB are not deleted in case you later want to recover the PDB.
 - You can also use Oracle Recovery Manager (RMAN) to delete archived logs and backups.
- You use the `DROP PLUGGABLE DATABASE` statement to drop a pluggable database (PDB).
 - Example:

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

ORACLE

You use the `DROP PLUGGABLE DATABASE` statement to drop a pluggable database (PDB).

- Specify `KEEP DATAFILES` to retain the data files associated with the PDB after the PDB is dropped. The temp file for the PDB is deleted because it is no longer needed. This is the default. Keeping data files may be useful in scenarios where a PDB that is unplugged from one CDB is plugged into another CDB, with both CDBs sharing storage devices.
- Specify `INCLUDING DATAFILES` to delete the data files associated with the PDB being dropped. The temp file for the PDB is also deleted.

Prerequisites for using the `DROP PLUGGABLE DATABASE` statement to drop PDBs include:

- You must be connected to a CDB.
- The current container must be the root, you must be authenticated `AS SYSDBA` or `AS SYSOPER`, and the `SYSDBA` or `SYSOPER` privilege must be either granted to you commonly, or granted to you locally in the root and locally in PDB you want to drop.
- To specify `KEEP DATAFILES` (the default), the PDB you want to drop must be unplugged.
- To specify `INCLUDING DATAFILES`, the PDB you want to drop must be in mounted mode or it must be unplugged.



View:

Review the following example in *Oracle Database SQL Language Reference*. This example shows you how to drop a PDB and all its data files.

- `DROP PLUGGABLE DATABASE` (Scroll down to "Dropping a PDB: Example")

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Summary for Lesson 3

In this lesson, you should have learned how to:

- Describe the methods and tools used to create PDBs
- Create PDBs from seed with SQL*Plus
- Clone PDBs with SQL*Plus
- Unplug and plug in PDBs with SQL*Plus
- Drop PDBs with SQL*Plus



ORACLE

Practice 3 Overview

- 3-1: Creating a PDB from Seed with SQL*Plus
- 3-2: Cloning a PDB with SQL*Plus
- 3-3: Unplugging and Plugging a PDB with SQL*Plus
- 3-4: Dropping a PDB

Practice 3-1 Creating a PDB from Seed

Overview

In this practice, you create an empty PDB named PDB2 in your CDB by using the seed PDB.

Note: You can use Database Configuration Assistant, SQL Developer, or SQL commands to create a PDB from seed. This practice shows you how to do it by using SQL commands in SQL*Plus.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Complete the following steps on VM1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Create a directory to store PDB2's data files. This step is an important prerequisite.

- a. Use the `mkdir` command to create the PDB2 directory. Make sure to use lower case for `mkdir`. No results are displayed if the command succeeds.

```
$ mkdir $ORACLE_BASE/oradata/ORCL/PDB2
$
```

- b. Verify that the PDB2 directory is created. The result lists the PDB2 directory. Your timestamp will differ.

```
$ ls -l /u01/app/oracle/oradata/ORCL | grep PDB2
drwxr-xr-x 2 oracle oinstall        4096 Jul 19 18:16 PDB2
$
```

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

Important! To create a PDB from seed, you must connect to the root container as a user with the `CREATE PLUGGABLE DATABASE` privilege.

```
$ sqlplus / as sysdba
```

```
SQL*Plus: Release 12.2.0.1.0 Production on Fri 15 21:27:43 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64-bit Production
SQL>
```

4. Create PDB2 by using the `CREATE PLUGGABLE DATABASE` command. Specify an admin user named PDBADMIN, and grant this user the DBA role. See Appendix - Product-Specific Credentials for the password. The seed PDB does not have a USERS data file and tablespace. Therefore, include the `DEFAULT TABLESPACE USERS` clause to create a default permanent tablespace for any non-administrative users for which you do not specify a different permanent tablespace. To specify the target location of the data files, include the `FILE_NAME_CONVERT` clause. This clause enables you to specify the target locations of the files based on the names of the source files. The first parameter in the clause is the source directory of the seed data files. The second is the destination directory for the new PDB data files.

```
SQL> CREATE PLUGGABLE DATABASE PDB2 ADMIN USER PDB2ADMIN
  2  IDENTIFIED BY <password>
  3  ROLES=(dba)
  4  DEFAULT TABLESPACE USERS
  5  DATAFILE '/u01/app/oracle/oradata/ORCL/PDB2/users01.dbf'
  6  SIZE 250M AUTOEXTEND ON
  7  FILE_NAME_CONVERT=(''/u01/app/oracle/oradata/ORCL/pdbseed/'',
  '"/u01/app/oracle/oradata/ORCL/PDB2/');
```

```
Pluggable database created.
SQL>
```

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

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

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

5. Open PDB2 by using the `ALTER PLUGGABLE DATABASE` command.

```
SQL> ALTER PLUGGABLE DATABASE PDB2 OPEN;  
Pluggable database altered.  
SQL>
```

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

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

6. View the list of services registered with the listener. When you create a PDB, a service is created and started inside the PDB. The name of the service is the same name as the PDB. In this case, the service name is `pdb2.example.com`, and it has one instance, as shown below. You will connect to this service in the next step.

Samad Ul Haq (samad.ulhaq@arcanainfo.com)
non-transferable license to use this Student Guide.

```

SQL> !lsnrctl status

LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 16-JUL-2016 11:38:12

Copyright (c) 1991, 2016, Oracle. All rights reserved.

Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
STATUS of the LISTENER
-----
Alias                      LISTENER1
Version                    TNSLSNR for Linux: Version 12.2.0.1.0 - Production
Start Date                 14-JUL-2016 18:03:46
Uptime                     1 days 17 hr. 34 min. 26 sec
Trace Level                off
Security                   ON: Local OS Authentication
SNMP                       OFF
Listener Parameter File   /u01/app/oracle/product/12.2.0/dbhome_1/network/admin/listener.ora
Listener Log File          /u01/app/oracle/diag/tnslsnr/12cr2db/listener1/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=12cr2db.us.oracle.com)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=12cr2db.us.oracle.com)(PORT=5500))

  (Security=(my_wallet_directory=/home/oracle/admin/ORCL/xdb_wallet))
    (Presentation=HTTP)(Session=RAW)
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=12cr2db.us.oracle.com)(PORT=5501))
    (Presentation=HTTP)(Session=RAW))
Services Summary...
Service "379d6befb4634ac8e0530d11ed0ad32d.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "37b460507162736de0530d11ed0a2417.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "ORCL.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "ORCLXDB.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "pdb1.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "pdb2.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
The command completed successfully
SQL>

```

7. Connect to PDB2 as the PDB2ADMIN user by using the Easy Connect method. See [Appendix - Product-Specific Credentials](#) for the password.

```

SQL> CONNECT PDB2ADMIN/<password>@localhost:1521/pdb2.example.com

Connected.
SQL>

```

Note: Alternatively, you could have switched to PDB2 by using the ALTER SESSION command. This command connects you to PDB2 as the SYS user.

```
SQL> ALTER SESSION SET container = PDB2;
Session altered.
SQL>
```

8. Explore PDB2.

- Show the current container. It is PDB2.

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

- Show the current container ID. It is 4.

```
SQL> SHOW con_id
CON_ID
-----
4
SQL>
```

- List the service for PDB2 by querying the V\$SERVICES view. The query returns the pdb2.example.com service.

```
SQL> COLUMN name FORMAT A20
SQL> SELECT name FROM v$services;
NAME
-----
pdb2.example.com
SQL>
```

- List the data files for PDB2 and their respective tablespaces by querying the DBA_DATA_FILES view. The query returns four data files. The results below are formatted for easier viewing.

```
SQL> SELECT file_name, tablespace_name FROM dba_data_files;
FILE_NAME                                TABLESPACE
-----
/u01/app/oracle/oradata/ORCL/PDB2/system01.dbf    SYSTEM
/u01/app/oracle/oradata/ORCL/PDB2/sysaux01.dbf   SYSAUX
/u01/app/oracle/oradata/ORCL/PDB2/undotbs01.dbf  UNDOTBS1
/u01/app/oracle/oradata/ORCL/PDB2/users01.pdf    USERS
SQL>
```

- e. List the temp files for PDB2 by querying the DBA_TEMP_FILES view. The query returns one temp file. Your temp file name will be different than the one shown below. The results are formatted for easier viewing.

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

FILE_NAME                                TABLESPACE
-----
/u01/app/oracle/oradata/ORCL/PDB2/
temp012016-07-14_18-07-01-801-PM.dbf      TEMP
SQL>
```

- f. List the local users for PDB2 by querying the DBA_USERS view. The query returns PDB2_ADMIN, which is the user you specified when creating the PDB.

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

USERNAME
-----
PDB2ADMIN
SQL>
```

- g. List the common users for PDB2 by querying the DBA_USERS view. The query returns 36 users.

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

USERNAME
-----
DVF
DBSFUSER
XS$NULL
MDSYS
DIP
SYSKM
OUTLN
CTXSYS
OLAPSYS
SYSBACKUP
GSMUSER

USERNAME
-----
SPATIAL_CSW_ADMIN_USR
SYSTEM
DVSYS
SYSRAC
AUDSYS
ORACLE_OCM
DBSNMP
GGSYS
GSMADMIN_INTERNAL
ORDPLUGINS
ORDSYS

USERNAME
-----
MDDATA
OJVMSYS
SYS$UMF
APPQOSSYS
ORDDATA
XDB
GSMCATUSER
SYSDG
SYS
LBACSYS
ANONYMOUS

USERNAME
-----
SI_INFORMTN_SCHEMA
WMSYS
REMOTE_SCHEDULER_AGENT

36 rows selected.
SQL>
```

- h. Exit SQL*Plus and close the terminal window.

```
SQL > EXIT  
.  
$
```

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 3-2 Hot Cloning a PDB

Overview

In this practice, you use SQL*Plus to hot clone PDB1 as PDB3 in the CDB.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Complete the following steps on VM1.

Unlock the HR Account in Window 1

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This terminal window will be referred to as window 1.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Create a directory for the new PDB, which we'll name `PDB3`, under the CDB file location. If the subdirectory is created successfully, no results are returned.

```
$ mkdir $ORACLE_BASE/oradata/ORCL/PDB3
$
```

3. Start SQL*Plus and connect to the root container as the `SYS` user with the `SYSDBA` privilege. You must create the clone from within the root container.

The `SYS` user also has the `CREATE PLUGGABLE DATABASE` privilege, which is required to clone a PDB.

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Fri Jul 15 19:17:37 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

4. Switch to `PDB1`.

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

Session altered.
SQL>
```

5. Unlock the hr account and set the password. See Appendix - Product-Specific Credentials for the password.

```
SQL> ALTER USER hr ACCOUNT UNLOCK IDENTIFIED BY <password>;
User altered.
SQL>
```

6. Switch back to CDB\$ROOT.

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;
Session altered.
SQL>
```

Start a Transaction in PDB1 (Window 2)

1. Start a transaction in PDB1 that will not be entirely completed before you clone PDB1.
 - a. Open a new terminal window and source the oraenv script. This window will be referred to as window 2.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

- b. Start SQL*Plus and connect to PDB1 as the hr user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus hr/<password>@PDB1

SQL*Plus: Release 12.2.0.1.0 Production on Fri Jul 15 19:17:37 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
SQL>
```

- c. Display the salary for employee ID 100.

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

SALARY
-----
24000
SQL>
```

- d. Update the EMPLOYEES table so that the employee salaries are increased by 10%. Note: The update will not be completed until you "commit" the transaction. However, you will commit this transaction after you clone PDB1.

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

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

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

SALARY
-----
26400
SQL>
```

Clone PDB1 as PDB3 in Window 1

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

1. In window 1, create a clone named PDB3 from PDB1 by using the `CREATE PLUGGABLE DATABASE` statement. This step takes about ten seconds to complete.

```
SQL> CREATE PLUGGABLE DATABASE PDB3 FROM PDB1
  2  CREATE_FILE_DEST= '/u01/app/oracle/oradata/ORCL/PDB3';

Pluggable database created.
SQL>
```

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

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

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

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

```
SQL> ALTER PLUGGABLE DATABASE PDB3 OPEN;

Pluggable database altered.
SQL>
```

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

```
SQL> SHOW PDBS
```

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

```
SQL>
```

Commit the Transaction in Window 2

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

```
SQL> COMMIT;
```

```
Commit complete.  
SQL>
```

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

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

```
SALARY  
-----  
26400  
SQL>
```

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

Answer: Continue to the next section to find out.

Explore PDB3 in Window 1

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

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

```
Session altered.  
SQL>
```

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

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

```
SALARY  
-----  
24000  
SQL>
```

3. Question: The original salary was 24000. Earlier in window 2, you updated the salary to 26400. Why isn't the salary showing as 26400 now?
 Answer: The salary was not increased because the COMMIT statement took place *after* the clone operation had completed.
4. List the service for PDB3 by querying the V\$SERVICES view. The query returns the pdb3.example.com service.

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

NAME
-----
pdb3.example.com
SQL>
```

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

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

FILE_NAME                                TABLESPACE
-----
/u01/app/oracle/oradata/ORCL/PDB3/ORCL/
37F073AA023D7F20E0530D11ED0A8ECE/datafile/
o1_mf_users_crtkof0f_.dbf                USERS
/u01/app/oracle/oradata/ORCL/PDB3/ORCL/
37F073AA023D7F20E0530D11ED0A8ECE/datafile/
o1_mf_undotbs1_crtkof0d_.dbf              UNDOTBS1
/u01/app/oracle/oradata/ORCL/PDB3/ORCL/
37F073AA023D7F20E0530D11ED0A8ECE/datafile/
o1_mf_sysaux_crtkof0c_.dbf                SYSAUX
/u01/app/oracle/oradata/ORCL/PDB3/ORCL/
37F073AA023D7F20E0530D11ED0A8ECE/datafile/
o1_mf_system_crtkof01_.dbf                SYSTEM
SQL>
```

6. Question: Do you notice a difference between the data file names in the previous step versus the names created in the previous practice when you created a PDB from seed?
 Answer: In this case, Oracle Managed Files (OMF) names the data files for you because you used the CREATE_FILE_DEST clause, which only defines the directory for the data files. This clause comes from the initialization parameter DB_CREATE_FILE_DEST. If you use this parameter, then all your PDB data files will end up in the same directory; whereas, using the CREATE_FILE_DEST clause enables you to specify distinct directories for each PDB.
7. List the temp file(s) for PDB3 by querying the DBA_TEMP_FILES view. The query returns one temp file. The name of your temp file will be different than the one shown below.

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

FILE_NAME	TABLESPACE
/u01/app/oracle/oradata/ORCL/PDB3/ORCL/ 37F073AA023D7F20E0530D11ED0A8ECE/datafile/ o1_mf_temp_crtkof0d_.dbf	TEMP

8. List the local users for PDB3 by querying the DBA_USERS view. The query returns the PDBADMIN and HR users.

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

USERNAME
PDBADMIN
HR

9. List the common users for PDB3 by querying the DBA_USERS view. The query returns 36 users.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

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

USERNAME
-----
DVF
DBSFUSER
XS$NULL
MDSYS
DIP
SYSKM
OUTLN
CTXSYS
OLAPSYS
SYSBACKUP
GSMUSER
SPATIAL_CSW_ADMIN_USR
SYSTEM
DVSYN
SYSRAC
AUDSYS
ORACLE_OCM
DBSNMP
GGSYS
GSMADMIN_INTERNAL
ORDPLUGINS
ORDSYS
MDDATA
OJVMSYS
SYS$UMF
APPQOSSYS
ORDDATA
XDB
GSMCATUSER
SYSDG
SYS
LBACSYS
ANONYMOUS
SI_INFORMTN_SCHEMA
WMSYS
REMOTE_SCHEDULER_AGENT

36 rows selected.

SQL>
```

Return Salary Values to Their Original Values (Window 2)

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

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

3. Commit the transaction.

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

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

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

5. Exit SQL*Plus in both window 1 and window 2 and close both windows.

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

Practice 3-3 Unplugging and Plugging a PDB

Overview

In this practice, you unplug PDB3 from the ORCL CDB and plug it back into ORCL CDB. You give the PDB a new name (HRPDB) when you plug it back in.

Assumptions

You are logged in to VM1 as the `oracle` user.

You completed [Practice 3-2 Hot Cloning a PDB](#).

Tasks

Complete the following steps on VM1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Unplug PDB3 in the ORCL CDB.

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

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Wed Jul 20 19:40:40 2016

Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
SQL>
```

- b. Close PDB3.

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

```
SQL> ALTER PLUGGABLE DATABASE PDB3 CLOSE IMMEDIATE;

Pluggable database altered.
SQL>
```

- c. Unplug PDB3 into an XML file (`/u01/app/oracle/oradata/pdb3.xml`).

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

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

```
SQL> ALTER PLUGGABLE DATABASE PDB3 UNPLUG INTO
  '/u01/app/oracle/oradata/PDB3.xml';

```

```
Pluggable database altered.
SQL>
```

- d. Drop PDB3 while it is closed, but keep its datafiles.

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

```
Pluggable database dropped.
SQL>
```

- e. Verify the status of the unplugged PDB3 by querying the CDB_PDBS view. No rows are returned by the query.

```
SQL> SELECT pdb_name, status FROM cdb_pdbs WHERE pdb_name IN ('PDB3');
```

```
no rows selected
SQL>
```

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

- a. Make sure that PDB3 is compatible with the ORCL CDB. Execution of the following PL/SQL block raises an error if it is not compatible. Tip: Enter each line, followed by a return, and the whole procedure will run after you close with a slash.

```
SQL> set serveroutput on
SQL> DECLARE
  2  compatible BOOLEAN := FALSE;
  3  BEGIN
  4  compatible := DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
  5    pdb_descr_file => '/u01/app/oracle/oradata/PDB3.xml');
  6  if compatible then
  7    DBMS_OUTPUT.PUT_LINE('PDB3 is compatible');
  8  else DBMS_OUTPUT.PUT_LINE('PDB3 is not compatible');
  9  end if;
 10 END;
11 /
```

```
PDB3 is compatible
PL/SQL procedure successfully completed.
SQL>
```

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

This operation lasts a few seconds. The original data files of the unplugged PDB now belong to the new plugged-in PDB.

```
SQL> CREATE PLUGGABLE DATABASE HRPDB USING
  '/u01/app/oracle/oradata/PDB3.xml'  NOCOPY  TEMPFILE REUSE;
Pluggable database created.
SQL>
```

4. Examine the plugged-in PDB.

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

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

NAME          CON_ID
-----        -----
CDB$ROOT      1
PDB$SEED      2
PDB1          3
PDB2          4
HRPDB         5
SQL>
```

- Show the status of HRPDB by querying the CDB_PDBS view. The results below are formatted for easier viewing.

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

PDB_NAME          STATUS
-----            -----
HRPDB             NEW
SQL>
```

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

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

OPEN_MODE
-----
MOUNTED
SQL>
```

- List the data files of HRPDB by querying the V\$DATAFILE view. Recall that the HRPDB container's ID is equal to five. Your paths and data file names will be different than those shown below.

```
SQL> COLUMN name FORMAT A50
SQL> SELECT name FROM v$logfile WHERE con_id=5;

NAME
-----
/u01/app/oracle/oradata/ORCL/PDB3/ORCL/44D3BFCAFE4
05C57E0534A10ED0A31B0/datafile/o1_mf_system_d6bzxm
m3_.dbf

/u01/app/oracle/oradata/ORCL/PDB3/ORCL/44D3BFCAFE4
05C57E0534A10ED0A31B0/datafile/o1_mf_sysaux_d6bzxm
mk_.dbf

/u01/app/oracle/oradata/ORCL/PDB3/ORCL/44D3BFCAFE4
05C57E0534A10ED0A31B0/datafile/o1_mf_undotbs1_d6bz
xmml_.dbf

/u01/app/oracle/oradata/ORCL/PDB3/ORCL/44D3BFCAFE4
05C57E0534A10ED0A31B0/datafile/o1_mf_users_d6bzxm
m_.dbf
SQL>
```

5. Open and connect to HRPDB.

a. Open HRPDB.

```
SQL> ALTER PLUGGABLE DATABASE HRPDB open;
Pluggable database altered.
SQL>
```

b. Connect to HRPDB as the SYS user with the SYSDBA privilege. See [Appendix D - Product-Specific Credentials](#) for the password.

```
SQL> CONNECT SYS/<password>@localhost:1521/HRPDB.example.com AS SYSDBA
Connected.
SQL>
```

6. Verify the current container name is HRPDB.

```
SQL> SHOW con_name
CON_NAME
-----
HRPDB
SQL>
```

7. Exit from SQL*Plus and close the terminal window.

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

Practice 3-4 Dropping a PDB

Overview

In this practice, you drop the PDB named `HRPDB` with SQL*Plus. Keep in mind that you could use a variety of tools to accomplish this task.

When you're finished, your VM1 machine should have one CDB named `ORCL` with two PDBs named `PDB1` and `PDB2`. `PDB1` has the `HR` sample data whereas `PDB2` does not.

Assumptions

You are logged in to VM1 as the `oracle` user.

You completed the following practices in this lesson:

- Practice 3-1 Creating a PDB from Seed
- Practice 3-2 Hot Cloning a PDB

Tasks

Complete the following steps on VM1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

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

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

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

```
SQL> SHOW PDBS
```

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

```
SQL>
```

4. Close HRPDB.

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

Pluggable database altered.

```
SQL>
```

5. Drop HRPDB, including its data files, by using the `DROP PLUGGABLE DATABASE` statement.

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

Pluggable database dropped.

```
SQL>
```

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

```
SQL> SHOW PDBS
```

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

```
SQL>
```

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

```
SQL> EXIT
```

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
\$

4

Configuring the Oracle Network Environment



ORACLE®

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Objectives for Lesson 4

After completing this lesson, you should be able to:

- Describe Oracle Net Services
- Explain how listeners work
- Configure listeners for dynamic or static service registration
- Configure local naming for database connections
- Test Oracle Net connectivity with `tnsping`
- Configure communication between databases by creating database links
- Explain the difference between dedicated and shared server configurations



ORACLE

Oracle Net Services

- Oracle Net Services gets installed with Oracle Database.
 - Its most common use is to enable network connections from client or middle-tier applications to an Oracle database server.
 - It provides network connections between multiple database instances.
 - It enables you to configure additional net services to allow access to external code libraries (EXTPROC).
 - It enables you to connect database instances to non-Oracle data sources (such as Sybase) through Oracle Heterogeneous Services.
- Oracle Net is a suite of networking components.
 - Oracle Net
 - Oracle Net Listener
 - Oracle Net Manager
 - Oracle Connection Manager
 - Oracle Net Configuration Assistant
- You can use EMCC and Listener Control utility to manage Oracle Net Services.

ORACLE®

Oracle Net

Oracle Net is communication software that enables a network session from a client application to an Oracle database server. After it establishes a network session, Oracle Net acts as the data courier for both the client application and the database server. It's responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. Oracle Net (or something that simulates Oracle Net, such as Java Database Connectivity) is installed, by default, on the database server, however you must install it separately on client computers that need to talk to the database server. On a client computer, Oracle Net operates as a background component for application connections to the database server.

Oracle Net Listener

Oracle Net Listener is an active process on the database server, included with Oracle Net, whose responsibility is to listen for incoming client connection requests and manage the traffic to the database server. Think of it as the gateway to the database instance for all non-local user connections. You can use Enterprise Manager Cloud Control or Oracle Net Manager to configure listeners.

Oracle Net Manager

Oracle Net Manager is an Oracle Net tool for configuring Oracle Net Services for an Oracle home on a local client or server host. Post installation, you can configure listeners, naming, naming methods, and profiles.

You can invoke Oracle Net Manager in the following ways:

- On Linux, enter `netmgr` at the operating system prompt.
- On Microsoft Windows, select **Programs** from the Start menu, then **Oracle - <home name>**, then **Configuration and Migration Tools**, and then **Net Manager**.

Oracle Connection Manager

Oracle Connection Manager is an optional proxy server, usually residing on a computer separate from client and database server computers, that enables you to configure rule-based access control for users and session multiplexing.

Oracle Net Configuration Assistant

Oracle Net Configuration Assistant (netca) is a stand-alone Oracle Net tool for configuring basic network components:

- Listener names and protocol addresses
- Naming methods
- Net service names
- Directory server usage: Configures a directory server for directory-enabled features

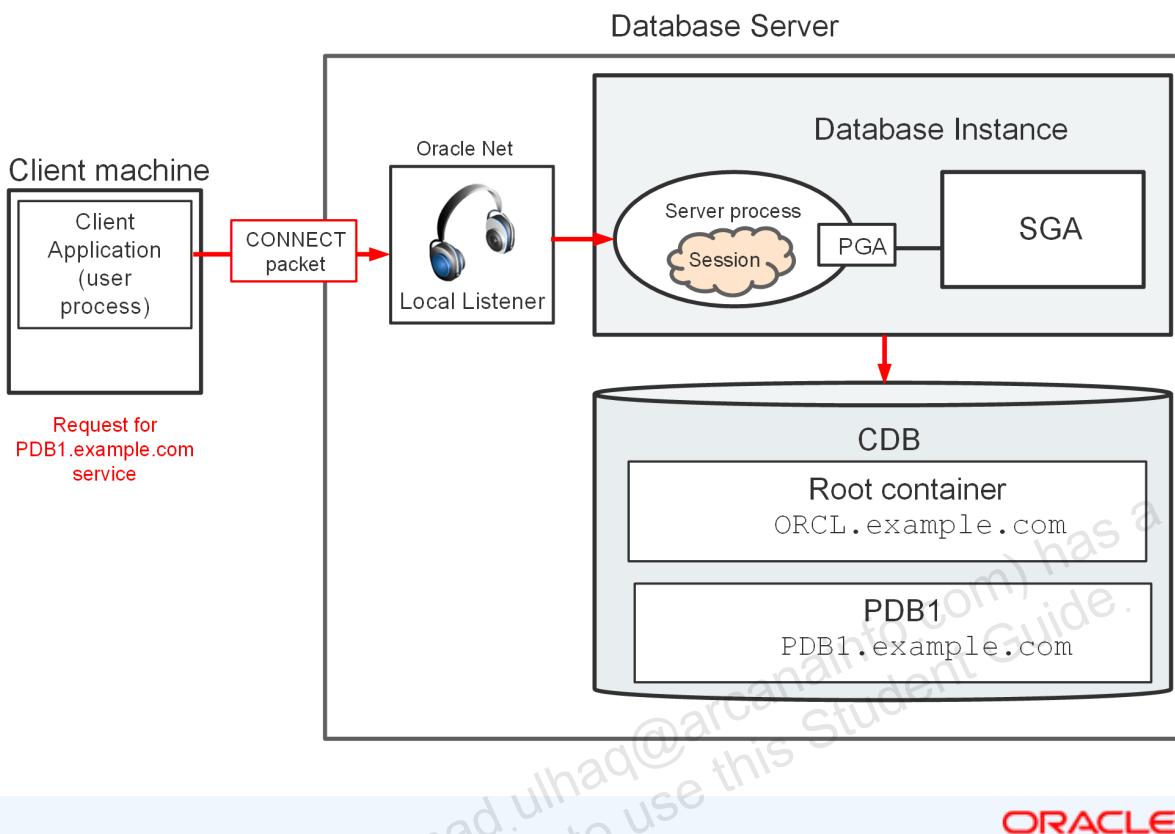
Oracle Net Configuration Assistant is executed by the Oracle Universal Installer during installation of Oracle Database software.

Other Tools

In addition to Oracle Net Manager and Oracle Net Configuration Assistant, you can use Enterprise Manager Cloud Control and the Listener Control utility to manage Oracle Net Services:

- **Enterprise Manager Cloud Control** (EMCC) is a large-scale web application that lets you configure Oracle Net Services for any Oracle home across multiple file systems. You can configure listeners, naming (net service names), naming methods, and profiles.
- The **Listener Control** utility is a tool that enables you to start, stop, check the status of a listener, reinitialize a listener from the configuration file parameters, dynamically configure many listeners, and change the listener password.

How Listeners Work



About Listeners

A *listener* is a separate process that runs locally on the database server (that is, on the same machine as the database instance) or remotely on a different database server. It receives incoming client connection requests and manages the traffic of these requests to the database server. Remote listeners are typically used in Oracle RAC environments.

A listener is configured with the following elements:

- A name
- A protocol address on which it listens for client requests (protocol, host name, and port number)
- A list of database services that it supports (for example, ORCL.example.com and PDB1.example.com)
- Control parameters that control the behavior of the listener

The Process of Establishing a Connection and Session

The first point of contact that a client application (hereafter referred to as the user process) makes is with an Oracle Net Listener (or simply, a listener). The listener is an active process included with Oracle Net software. It can reside locally on the same machine as the database instance or remotely, on its own computer (for Oracle RAC configurations). The process of establishing a connection to a database instance from a client machine and establishing a session is described as follows and is illustrated above.

1. The listener receives a CONNECT packet from the user process and verifies whether the service name is valid

in the packet.

2. If the service name is valid, the listener spawns a new dedicated server process in the database instance to deal with the connection. If the service name isn't valid, the listener returns an error to the user process.
3. The listener connects to the server process and passes the initialization information to it, including the address information for the user process. At this point, the listener no longer deals with the connection and all work is passed to the server process.
4. The server process checks the user's authentication credentials (usually a password), and if the credentials are valid, the server process creates a user session in the database instance memory.
5. With the session established, the server process now acts as the user's agent on the server. The server process is responsible for:
 - Parsing and running any SQL statements issued through the application
 - Checking the database buffer cache for data blocks required to perform SQL statements
 - Reading necessary data blocks from data files on the disk into the database buffer cache portion of the System Global Area (SGA), if the blocks are not already present in the SGA
 - Managing all sorting activity. The Sort Area is a memory area that is used to work with sorting; it is contained in a portion of memory that is associated with the Program Global Area (PGA)
 - Returning results to the user process in such a way that the application can process the information
 - Reading auditing options and reporting user processes to the audit destination

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

The Default Listener

- During an Oracle Database installation, Oracle Universal Installer launches Oracle Net Configuration Assistant and creates a local listener named LISTENER.
 - LISTENER is automatically populated with available database services through a feature called dynamic service registration.
 - LISTENER listens on the following TCP/IP protocol address:

```
ADDRESS=(PROTOCOL=tcp)(HOST=host_name)(PORT=1521)
```
- Without any configuration, you can access your database instance immediately through LISTENER.
- If the listener name is LISTENER and it cannot be resolved, a protocol address of TCP/IP, and a port number of 1521 is assumed.

Tools to Configure Listeners

- You can use the following tools to create and manage listeners:
 - Oracle Net Manager
 - Oracle Net Configuration Assistant (netca)
 - Listener Control utility
 - Enterprise Manager Cloud Control
 - Database Configuration Assistant (allows you to create a listener when you create a CDB)
- The Listener Control utility connects only to listeners. It gets listener protocol addresses from:
 - `listener.ora` or
 - `tnsnames.ora` (naming method)

ORACLE

For a majority of commands, the Listener Control utility establishes an Oracle Net connection with the listener that is used to transmit the command. To initiate an Oracle Net connection to the listener, the Listener Control utility needs to obtain the protocol address for the named listener or default listener (`LISTENER`) if no name is specified. This is done by resolving the listener name with one of the following mechanisms:

- `listener.ora` file in the directory specified by the `TNS_ADMIN` environment variable
- `listener.ora` file in the `$ORACLE_HOME/network/admin` directory
- Naming method, for example, a `tnsnames.ora` file

Configuring Dynamic Service Registration

- By default, an Oracle database is configured to use dynamic service registration (service registration), which allows the Oracle database to identify its available services to listeners automatically.
- The LREG process polls the listeners to see if they're running, and if so, registers database service information to them.
- Dynamic service registration registers, by default, all PDB services to the same listener.
 - If you stop that listener, you stop access to all the PDB services.
- General steps to configure dynamic service registration:
 1. Make sure that the `INSTANCE_NAME`, `LOCAL_LISTENER`, `REMOTE_LISTENER`, and `SERVICE_NAMES` initialization parameters are properly configured.
 2. Configure protocol addresses (end points) in the server-side `tnsnames.ora` file.

ORACLE®

Benefits of Dynamic Service Registration

Service registration offers the following benefits:

- Connect-time failover: Because the listener always monitors the state of the instances, service registration facilitates automatic failover of a client connect request to a different instance if one instance is down.
- Connection load balancing: Service registration enables the listener to forward client connect requests to the least-loaded instance and dispatcher or dedicated server. Service registration balances the load across the service handlers and nodes.
- High-availability for Oracle Real Application Clusters and Oracle Data Guard

The Role of the LREG Process

The Listener Registration (LREG) process polls the listeners to see if they're running, and if so, registers the following database service information to them:

- Database instance name
- Database service names available on the database instance (for example, `ORCL.example.com` and `PDB1.example.com`)
- Current and maximum load for the database instance
- Service handlers (dispatchers and dedicated servers) available to the database instance

LREG registers with the listeners after the database instance mounts the database, and every 60 seconds afterwards.

How to Configure Dynamic Service Registration

The LREG process learns of the available listeners through the `LOCAL_LISTENER` and `REMOTE_LISTENER` parameters. These parameters specify listener alias names for local listeners and remote listeners. Both parameters can have multiple values. These aliases resolve to protocol addresses (end points) in the server-side `tnsnames.ora` file. Note: Clients can also have a `tnsnames.ora` file, which you'll learn about in a later lesson. Through dynamic service registration, the LREG process is then able to pass on information about the available database services to *all* listeners on those end points.

For example, suppose `LOCAL_LISTENER = LISTENER_HOST1` and `REMOTE_LISTENER = LISTENER_HOST2`.

In the `tnsnames.ora` file, the `LISTENER_HOST1` and `LISTENER_HOST2` aliases are resolved to two different end points on two different machines. Notice that the `CONNECT_DATA` section is not included.

```
LISTENER_HOST1 =
  (ADDRESS = (PROTOCOL = TCP)(HOST = host1.example.com)(PORT = 1521))
LISTENER_HOST2 =
  (ADDRESS = (PROTOCOL = TCP)(HOST = host2.example.com)(PORT = 1521))
```

For dynamic service registration to work properly, make sure that the `INSTANCE_NAME`, `LOCAL_LISTENER`, `REMOTE_LISTENER`, and `SERVICE_NAMES` parameters are configured properly. By default, the installer populates the `SERVICE_NAME` parameter with the global database name, for example, `ORCL.example.com`, which provides one database service name that users can use to access the database instance. You can specify multiple service names for the database instance; however, if you want to distinguish among different uses of the same database, Oracle Database Resource Manager lets you view information about the user activity for each service name.

Dynamic service registration does not use the `listener.ora` file.

Configuring Static Service Registration

- Static service registration is a method for configuring listeners to obtain their service information manually.
 - You can create a listener for a particular PDB.
 - Static service registration might be required for some services, such as external procedures and heterogeneous services (for non-Oracle systems).
- With static registration, the listener has no knowledge of whether its database services exist or not. It only knows that it supports them.
 - The Listener Configuration utility shows the services status as UNKNOWN.
- You can have both static listeners and dynamic listeners configured at the same time.
- General steps to configure static service registration:
 1. In `listener.ora`, define a listener and its protocol addresses.
 2. In `listener.ora`, also create a `SID_LIST_<listener_name>` section that lists the database services for the listener.

ORACLE

Advantages of Static Service Registration

The following are some advantages to using static service registration:

- Static service registration enables you to create a listener for a particular PDB.
- Sometimes you may need the database instance up and running without anyone being able to log in. As soon as it is started up, dynamic service registration will automatically start registering all the database services to the listener, making the database instance available to users.
- There is also a difference in error messages returned between a static listener (which can point to a database service that is down) and a dynamic listener entry (which shows non-existence) when the database instance is shut down. The first case knows about the database service's existence and gives you an error message with useful information. The second case has no information and can't distinguish between a typo you may have made in the service name and whether it actually even exists.

How To Configure Static Service Registration

To create a static listener, you configure the `listener.ora` file. In that file, you define two sections. First, define the listener and its protocol addresses. Second, create a `SID_LIST_<listener_name>` section that lists the database services for the listener. For each service, include the following parameters:

- `GLOBAL_DBNAME` - equals the PDB's service name, for example, `PDB1.example.com`
- `ORACLE_HOME` - equals the Oracle home directory, for example,
`/u01/app/oracle/product/12.2.0/dbhome_1`

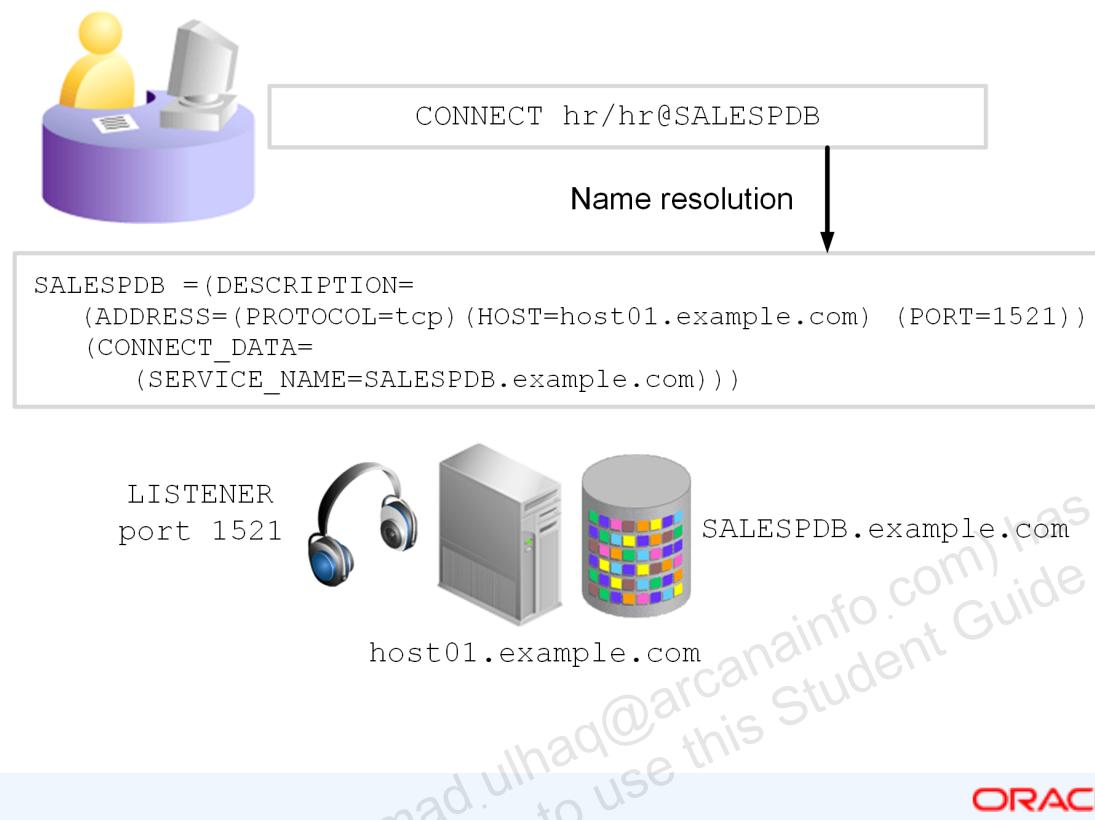
- SID_NAME - equals the name of your database instance, for example, ORCL

By default, the `listener.ora` file is stored in the `$ORACLE_HOME/network/admin` directory on the database instance machine.

```
LISTENER_SALESPDBS =
(DESCRIPTION_LIST =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = host1.example.com)(PORT = 1561)))
 )

SID_LIST_LISTENER_SALESPDBS =
(SID_LIST =
 (SID_DESC =
  (GLOBAL_DBNAME = PDB1.example.com)
  (SID_NAME = ORCL)
  (ORACLE_HOME = /u01/app/oracle/product/12.2.0/dbhome_1)
 )
 (SID_DESC =
  (GLOBAL_DBNAME = PDB2.example.com)
  (SID_NAME = ORCL)
  (ORACLE_HOME = /u01/app/oracle/product/12.2.0/dbhome_1)
 )
)
```

Configuring Local Naming for Connections



Local Naming Method

The *local naming method* is a way to store database connection information locally on a client machine. Local naming is appropriate for organizations in which Oracle Net service configurations do not change often.

Remember in the [Connecting to Oracle Databases](#) topic in Lesson 1 when you learned how to use the Easy Connect syntax to connect to a database service? Remember how you had to type all the connection information about the listener and database service in a connect string to make a direct connection to a database service, for example:

```
SQL> CONNECT hr/hr@host01.example.com:1521/SALESPDB.example.com
```

Well, rather than typing all that connection information in every time you want to get connected, you can put the information in a local file named `tnsnames.ora` on your client machine and give the connection a name (for example, `PDB1`). Then, when you connect, you simply use that name in your connect string (as shown below). Behind the scenes, Oracle Net resolves the name to the connection information in the `tnsnames.ora` file, and sends it off to the listener.

```
SQL> CONNECT hr/hr@SALESPDB
```

How to Configure Local Naming

To configure the local naming method, you add entries to a client-side `tnsnames.ora` file that map names (connect identifiers) to connection details (connect descriptors). This method is illustrated in the diagram above.

- **Connect identifier** (usually called net service name or net service alias): Choose a name to represent the database service to which you want to connect.
- **Connect descriptor**: Enter the address of the database listener, which includes the host on which the listener resides, the port on which the listener listens, the protocol that the listener accepts (for example, TCP/IP), and the name of the database service.

By default, the `tnsnames.ora` file is located in the `$ORACLE_HOME/network/admin` directory. You can locate it elsewhere using the `TNS_ADMIN` environment variable.

The following is an example entry in a `tnsnames.ora` file that makes the CONNECT statement above work. The net service name is `SALESPDB`. The listener resides on a machine named `host01.example.com`, the listener listens on port 1521, and the listener accepts TCP/IP protocol. The user will connect to the `SALESPDB.example.com` database service.

```
SALESPDB =
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=host01.example.com)(PORT=1521))
    (CONNECT_DATA=
      (SERVICE_NAME=SALESPDB.example.com)))
```

Other Naming Methods

In addition to using the Easy Connect and local naming methods, Oracle Net provides support for the following naming methods. These are not, however, practiced in this course.

- **Directory naming**: To access a database service, the directory naming method stores net service names in a centralized directory server that is compliant with the Lightweight Directory Access Protocol (LDAP).
- **External naming**: The external naming method stores net service names in a supported non-Oracle naming service. Supported third-party services include:
 - Network Information Service (NIS) External Naming
 - Distributed Computing Environment (DCE) Cell Directory Services (CDS)

Advanced Connection Options

- When a database service is accessible by multiple listener protocol addresses, you can specify the order in which the addresses are to be used.
- Oracle Net supports the following advanced connection options with local and directory naming:
 - Connect-time failover
 - Load balancing
 - Source routing

ORACLE®

Multiple Listener Protocol Addresses

When a database service is accessible by multiple listener protocol addresses, you can specify the order in which the addresses are to be used. The addresses can be chosen randomly or tried sequentially. In cases in which more than one listener is available, such as Oracle Real Application Clusters (RAC) configurations, Oracle Net can take advantage of listener failover and load balancing as well as Oracle Connection Manager source routing.

Connect-Time Failover

With *connect-time failover* enabled, the alias has two or more listener addresses listed. If the first address is not available, the second is tried. Oracle Net keeps trying addresses in the listed order until it reaches a listener that is functioning or until all addresses have been tried and failed. Transparent Application Failover (TAF) is a client-side feature that allows clients to reconnect to surviving databases in the event of a database instance failure. Notifications are used by the server to trigger TAF callbacks on the client side.

Load Balancing

With *load balancing* enabled, Oracle Net picks an address at random from the list of addresses. The runtime connection load-balancing feature improves connection performance by balancing the number of active connections among multiple dispatchers. In a RAC environment, connection pool load balancing also has the capability to balance the number of active connections among multiple instances.

Source Routing

Source routing is used with Oracle Connection Manager, which serves as a proxy server for Oracle Net traffic, enabling Oracle Net traffic to be routed securely through a firewall. Oracle Net treats the addresses as a list of relays, connecting to the first address and then requesting to be passed from the first to the second until the destination is reached. It differs from failover or load balancing in that all addresses are used each time a connection is made.

Testing Oracle Net Connectivity with tnsping

- The tnsping utility tests Oracle Net service aliases.
- It validates connectivity between a client and the Oracle Net Listener.
 - It validates that the host name, port, and protocol reach a listener.
 - It does not check whether the listener handles the service name.
 - It does not verify that the requested service is available.
- The tnsping utility also reveals the location of the configuration files.
 - In a system with multiple ORACLE_HOME locations, this can be helpful.
- Examples:
 - tnsping supports Easy Connect names resolution:

```
$tnsping host01.example.com:1521/orcl
```
 - tnsping also supports local and directory naming:

```
$ tnsping orcl
```

Configuring Communication Between Databases

- You must configure network connectivity (for example, `tnsnames.ora`) and a database link for database to database communication.
- A *database link* is a schema object that enables you to access objects on a different database.
- SQL command to create a database link:

```
SQL> CREATE DATABASE LINK <database_link_name>
```

- SQL command to connect to the other database:

```
SQL> CONNECT TO <user> IDENTIFIED BY <pwd> USING  
'<connect_string_for_remote_db>' ;
```

- Example SQL command to query a table on the other database:

```
SQL> SELECT * FROM employees@<database_link_name>;
```

ORACLE

About Database to Database Communication

To configure communication from your database to another database, you must configure the following on your database server:

- Network connectivity (for example, a net service name defined in the server-side `tnsnames.ora` file)
- Database link

A *database link* is a schema object that enables you to access objects on a different database.

- The other database need not be an Oracle database system.
- To access non-Oracle systems, you must use Oracle Heterogeneous Services.

When an application uses a database link to access a remote database, Oracle Database establishes a database session in the remote database on behalf of the local request.

You can create fixed user, current user, and connected user database links. Current user links are available only through the Oracle Advanced Security option. To create a private database link, you must have the `CREATE DATABASE LINK` system privilege. To create a public database link, you must have the `CREATE PUBLIC DATABASE LINK` system privilege.

Creating Database Links

The SQL command to create a database link is as follows. The CONNECT TO clause determines how the connection is established on the remote database. The <connect_string_for_remote_db> can be a net service name.

```
SQL> CREATE DATABASE LINK <database_link_name>
SQL> CONNECT TO <user> IDENTIFIED BY <pwd>
SQL> USING '<connect_string_for_remote_db>';
```

After you create a database link, you can use it to refer to tables and views on the other database. In SQL statements, you can refer to a table or view on the other database by appending @dblink to the table or view name. You can query a table or view on the other database or use any INSERT, UPDATE, DELETE, or LOCK TABLE statement for the table. For example:

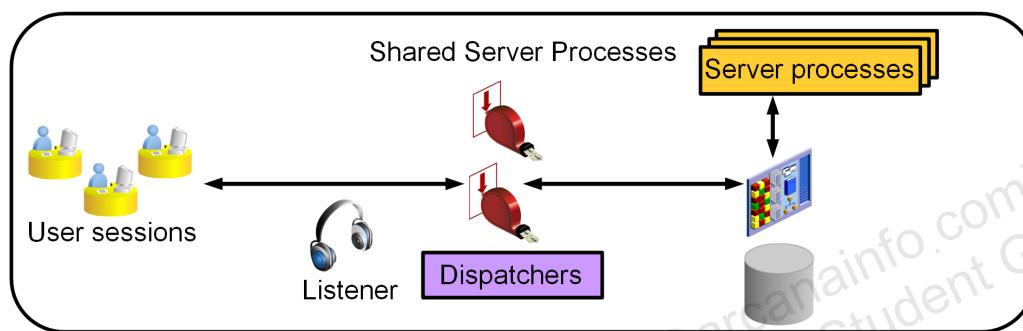
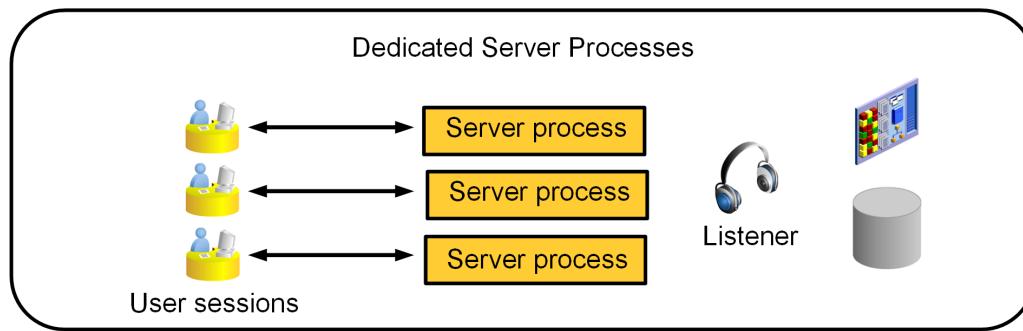
```
SQL> SELECT * FROM employees@<database_link_name>;
```

Listing Database Links

You can query the following views to determine which database links are defined in your database:

- dba_db_links: Lists all database links in the database
- all_db_links: Lists all database links accessible to the connected user
- user_db_links: Lists all database links owned by the connected user

Dedicated Versus Shared Server Configurations



ORACLE

Dedicated Server Configuration

In a *dedicated server configuration*, as illustrated in the diagram above, one server process handles requests for a single client process. Each server process uses system resources, including CPU cycles and memory. In a heavily loaded system, the memory and CPU resources that are used by dedicated server processes can be prohibitive and can negatively affect the system's scalability. If your system is being negatively affected by the resource demands of the dedicated server architecture, you have the following options:

- Increase system resources by adding more memory and additional CPU capability
- Use the Oracle Shared Server Process architecture

View:

Review the following example of a dedicated server configuration in *Oracle Database Administrator's Guide*. In this example, two distinct user processes are each connected to their own server process.

- **Dedicated Server Process** (see Figure 5-1 Oracle Database Dedicated Server Processes)

Shared Server Configuration

A *shared server configuration*, as illustrated in the diagram above, enables multiple client processes to share a small number of server processes. Each service that participates in the shared server process architecture has at least one *dispatcher process* (and usually more). When a connection request arrives, the listener does not spawn a dedicated server process. Instead, the listener maintains a list of dispatchers that are available for each service name, along with the connection load (number of concurrent connections) for each dispatcher. Connection requests are routed to the lightest loaded dispatcher that is servicing a given service name. Users remain connected to the same dispatcher for the duration of a session.

Unlike dedicated server processes, a single dispatcher can manage hundreds of user sessions. Dispatchers do not actually handle the work of user requests. Instead, they pass user requests to a common queue located in the shared pool portion of the SGA. Shared server processes take over most of the work of dedicated server processes, pulling requests from the queue and processing them until they are complete.

Because a user session may have requests processed by multiple shared server processes, most of the memory structures that are usually stored in the PGA must be in a shared memory location (by default, in the shared pool). However, if the large pool is configured or if `SGA_TARGET` is set for automatic memory management, these memory structures are stored in the large pool portion of the SGA.

The contents of the SGA and PGA is different in a shared server configuration than it is in a dedicated server configuration. In a shared server configuration:

- Text and parsed forms of all SQL statements are stored in the SGA.
- The cursor state contains runtime memory values for the SQL statement, such as rows retrieved.
- User-session data includes security and resource usage information.
- The stack space contains local variables for the process.

The change in the SGA and PGA is transparent to the user; however, if you are supporting multiple users, you need to increase the `LARGE_POOL_SIZE` initialization parameter. Each shared server process must access the data spaces of all sessions so that any server can handle requests from any session. Space is allocated in the SGA for each session's data space. You limit the amount of space that a session can allocate by setting the `PRIVATE_SGA` resource.

View:

Review the following example of a shared server configuration in *Oracle Database Administrator's Guide*. This example illustrates the order of events that occur when a client process accesses a database server with a shared server configuration.

- Shared Server Processes (see Figure 5-2 Oracle Database Shared Server Processes)

Pros and Cons

A dedicated server process is good for long-running queries and administrative tasks, and is faster than a shared server process in that there is always a server process ready to do work. However, an idle process or too many dedicated processes can result in an inefficient use of resources. Using shared server mode on the database server eliminates the need for a dedicated server process for each user connection, requires less memory for each user connection, and enables a larger number of users on a system with constrained memory. Dedicated server processes and shared server processes are enabled at the same time. Oracle XML DB (XDB) requires shared server processes and the Oracle database is already configured to use them. You will need to modify the initialization parameters for other users to use shared server processes.

The Oracle Shared Server architecture is an efficient process and memory use model, but it is not appropriate for all connections. Because of the common request queue and the fact that many users may share a dispatcher response queue, shared servers do not perform well with operations that must deal with large sets of data, such as warehouse queries or batch processing. Backup and recovery sessions that use Oracle Recovery Manager (discussed in later lessons) also deal with very large data sets and must use dedicated connections. Many administration tasks must not (and cannot) be performed by using shared server connections. These include starting up and shutting down the instance, creating tablespaces and data files, maintaining indexes and tables, analyzing statistics, and many other tasks that are commonly performed by the DBA. All DBA sessions must choose dedicated servers.

Summary for Lesson 4

Unauthorized reproduction or distribution prohibited
Copyright © 2017, Oracle and/or its affiliates.

In this lesson, you should have learned to:

- Describe Oracle Net Services
- Explain how listeners work
- Configure listeners for dynamic or static service registration
- Configure local naming for database connections
- Test Oracle Net connectivity with `tnsping`
- Configure communication between databases by creating database links
- Explain the difference between dedicated and shared server configurations



ORACLE

Samad Ul Haq (samad.ulhaq@arcanainfo.com)
non-transferable license to use this material

Practice 4 Overview

- 4-1: Exploring the Default Listener
- 4-2: Creating a Dynamic Listener
- 4-3: Creating a Static Listener
- 4-4: Creating a Net Service Name for a PDB
- 4-5: Creating a Database Link to an External Database

Note: The domain name written in the practices is example.com. On VM1, you will find a different domain name used, which is fine. This difference will not impact your practices.

Samad.Ulhaq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 4-1 Exploring the Default Listener

Overview

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

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Complete the following steps on VM1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

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

```
$ sqlplus / as sysdba
SQL*Plus: Release 12.2.0.1.0 Production on Fri 15 21:27:43 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64-bit Production
SQL>
```

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

- a. `INSTANCE_NAME`: This parameter identifies the database instance name. It defaults to the Oracle System Identifier (SID) of the database instance. The results show that the database instance name is `ORCL`, which you named during installation.

```
SQL> SHOW PARAMETER INSTANCE_NAME

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

- b. `SERVICE_NAMES`: This parameter identifies the service names that users can use in their connection strings to connect to the database instance. By default, the service name takes on the same name as the global database name, `ORCL.example.com`, which is a combination of the `DB_NAME` parameter (`ORCL`) and the `DB_DOMAIN` parameter (`example.com`). The `SERVICE_NAMES` parameter can accept multiple comma-separated values if you want to provide users with a variety of service names for the database instance. Doing so helps you control and monitor different user groups in Oracle Database Resource Manager.

```
SQL> SHOW PARAMETER SERVICE_NAMES
```

NAME	TYPE	VALUE
service_names	string	ORCL.example.com

- 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_ORCL` (`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_ORCL

- 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	

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

- a. Exit SQL*Plus.

```
SQL> EXIT
```

```
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0
- 64bit Production
$
```

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

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

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

```
$ ls

listener1608118PM3359.bak    shrept.lst
tnsnames1608118PM3359.bak
listener.ora                   sqlnet1608118PM3359.bak  tnsnames.ora
samples                         sqlnet.ora
$
```

- d. View the `tnsnames.ora` file by using the `cat` command (case matters). The entry for the `LISTENER_ORCL` alias contains one protocol address, which consists of a host name (`12cr2db.example.com`), port number (1521, which is the default port number), and protocol (TCP). The protocol address is the listener's "end point." A listener end point does not contain a listener name or a `CONNECT_DATA` section like the `PDB1` and `ORCL` entries.

```
$ cat tnsnames.ora

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

LISTENER_ORCL =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))

PDB1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PDB1.example.com)
    )
  )
ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL.example.com)
    )
  )
$
$
```

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:

- `listener.ora` file in the directory specified by the `TNS_ADMIN` environment variable. This is why it's important to set the environment variables to the appropriate home before using the Listener Control

utility, which you did at the beginning of this practice.

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

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

```
$ cat listener.ora

# listener.ora Network Configuration File:
/u01/app/oracle/product/12.2.0/dbhome_1/network/admin/listener.ora
# Generated by Oracle configuration tools.

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )
$
```

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.

```
$ lsnrctl

LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 18-AUG-2016 13:50:41
Copyright (c) 1991, 2016, 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 commands. The utility will automatically execute all commands against that listener. If you want to work on a different listener, you can either set the current_listener parameter to the other listener's name by using the

SET current_listener command, or you can include the other listener's name after each command. Currently, the default listener is set to LISTENER.

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

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

The alias name for the listener is the name you gave it during CDB 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. Some of your service names will be different than the ones shown below.

```

LSNRCTL> status

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP) (HOST=12cr2db.example.com)
(PORT=1521)))
STATUS of the LISTENER
-----
Alias                      LISTENER
Version                    TNSLSNR for Linux: Version 12.2.0.1.0 -
Production
Start Date                15-AUG-2016 19:57:39
Uptime                     2 days 1 hr. 25 min. 4 sec
Trace Level               off
Security                   ON: Local OS Authentication
SNMP                       OFF
Listener Parameter File   /u01/app/oracle/product/12.2.0/dbhome_1/network/
                           admin/listener.ora
Listener Log File         /u01/app/oracle/diag/tnslsnr/12cr2db/listener1/
                           alert/log.xml
Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=12cr2db.example.com)(PORT=1521)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=12cr2db.example.com)(PORT=5500))

(Security=(my_wallet_directory=/u01/app/oracle/admin/ORCL/xdb_wallet))
(Presentation=HTTP)(Session=RAW))
Services Summary...
Service "379d6befb4634ac8e0530d11ed0ad32d.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "38046a0b01822d55e0530d11ed0afe33.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "ORCL.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "ORCLXDB.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "pdb1.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service "pdb2.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
The command completed successfully
LSNRCTL>

```

- d. To view additional details about the registered services, issue the `services` command. The results tell you that there are six database services configured for the current listener, three of which are the CDB, PDB1, and PDB2 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.
 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.

```
LSNRCTL> services

Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
Services Summary...
Service "379d6befb4634ac8e0530d11ed0ad32d.example.com" has 1 instance(s).

  Instance "ORCL", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:1514 refused:0 state:ready
        LOCAL SERVER
Service "38046a0b01822d55e0530d11ed0afe33.example.com" has 1 instance(s).

  Instance "ORCL", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:1514 refused:0 state:ready
        LOCAL SERVER
Service "ORCL.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:1514 refused:0 state:ready
        LOCAL SERVER
Service "ORCLXDB.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
        DISPATCHER <machine: 12cr2db, pid: 6281>
          (ADDRESS=(PROTOCOL=tcp)(HOST=12cr2db.us.oracle.com)(PORT=39265))

Service "pdb1.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:1514 refused:0 state:ready
        LOCAL SERVER
Service "pdb2.example.com" has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:1514 refused:0 state:ready
        LOCAL SERVER
The command completed successfully
LSNRCTL>
```

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

```
LSNRCTL> show log_status

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=12cr2db.example.com)(PORT=1521)))
LISTENER parameter "log_status" set to ON
The command completed successfully
LSNRCTL>
```

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

```
LSNRCTL> show log_file

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=12cr2db.example.com)(PORT=1521)))
LISTENER parameter "log_file" set to /u01/app/oracle/diag/tnslsnr/12cr2db/
listener/alert/log.xml
The command completed successfully
LSNRCTL>
```

- g. Exit the Listener Control utility and close the terminal window.

```
LSNRCTL> exit
$
```

Practice 4-2 Creating a Dynamic Listener

Overview

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

Assumptions

You are logged in as the oracle user on VM1.

Tasks

Create an Entry in tnsnames.ora for LISTENER2

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

1. On your desktop, open **Computer**, and then **Filesystem**.
2. Browse to \$ORACLE_HOME/network/admin (
/u01/app/oracle/product/12.2.0/dbhome_1/network/admin).
3. Change the name of the tnsnames.ora file to tnsnames.ora.txt, and then open it in gedit.
4. Add an entry named LISTENER2 to resolve the alias to a protocol address, similar to the LISTENER_ORCL entry. You can copy and paste the LISTENER_ORCL as a starting point. Specify 12cr2db.example.com for the host name, 1561 for the port number, and TCP as the protocol.

Note: Your domain name for LISTENER_ORCL will be different than the one shown below.

```
# tnsnames.ora Network Configuration File:  
/u01/app/oracle/product/12.2.0/dbhome_1/network/admin/tnsnames.ora  
# Generated by Oracle configuration tools.  
  
LISTENER2 =  
  (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1561))  
  
LISTENER_ORCL =  
  (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com )(PORT = 1521))  
  
...
```

5. Save the file and then exit gedit.
6. Rename the file back to tnsnames.ora.
7. Close the "admin - File Browser" window.

Modify the LOCAL_LISTENER Initialization Parameter

Modify the LOCAL_LISTENER initialization parameter to include both LISTENER_ORCL and LISTENER2 aliases.

1. Open a new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL .

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

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

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Fri Aug 19 20:13:06 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

- View the LOCAL_LISTENER initialization parameter. The value LISTENER_ORCL 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_ORCL
SQL>
```

- 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 it's 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>
```

- Set the LOCAL_LISTENER parameter equal to LISTENER_ORCL and LISTENER2 by using the ALTER SYSTEM command. Specify SCOPE=BOTH to apply the change to both the server parameter file and memory. The change is made to the current instance and is effective immediately. Don't forget to enclose the listener aliases in double quotation marks.

```
SQL> ALTER SYSTEM SET local_listener="LISTENER_ORCL,LISTENER2" SCOPE=BOTH;

System altered.
SQL>
```

- Confirm that LISTENER_ORCL and LISTENER2 are values for the LOCAL_LISTENER initialization parameter.

```
SQL> SHOW PARAMETER local_listener
```

NAME	TYPE	VALUE
local_listener	string	LISTENER_ORCL,LISTENER2

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

```
SQL> EXIT
```

```
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

Add LISTENER2 to listener.ora

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. This section shows you how to add the entry two ways. You can either manually enter the listener information into the `listener.ora` file by using a text editor. Or, you can use Oracle Net Manager if you have access to Oracle Net Manager. It's important to remember that dynamic service registration does not make use of the `listener.ora` file; however, you do need to configure the file if you want to manage listeners with the Listener Control utility.

1. Complete this step if you want to manually configure the `listener.ora` file. Alternatively, skip this step and follow step 2, which uses Oracle Net Manager to configure the `listener.ora` file.

- a. On your desktop, open **Computer**, and then **Filesystem**.
- b. Browse to `$ORACLE_HOME/network/admin` (`/u01/app/oracle/product/12.2.0/dbhome_1/network/admin`).
- c. Change the name of the `listener.ora` file to `listener.ora.txt`, and then open it in gedit.
- d. Add the following code to the top of the file, right after the first two comments. Be sure to enter `12cr2db.example.com` for the host name.

Important: Do not use copy/paste to add this text. You must type the text or you will get errors later on.

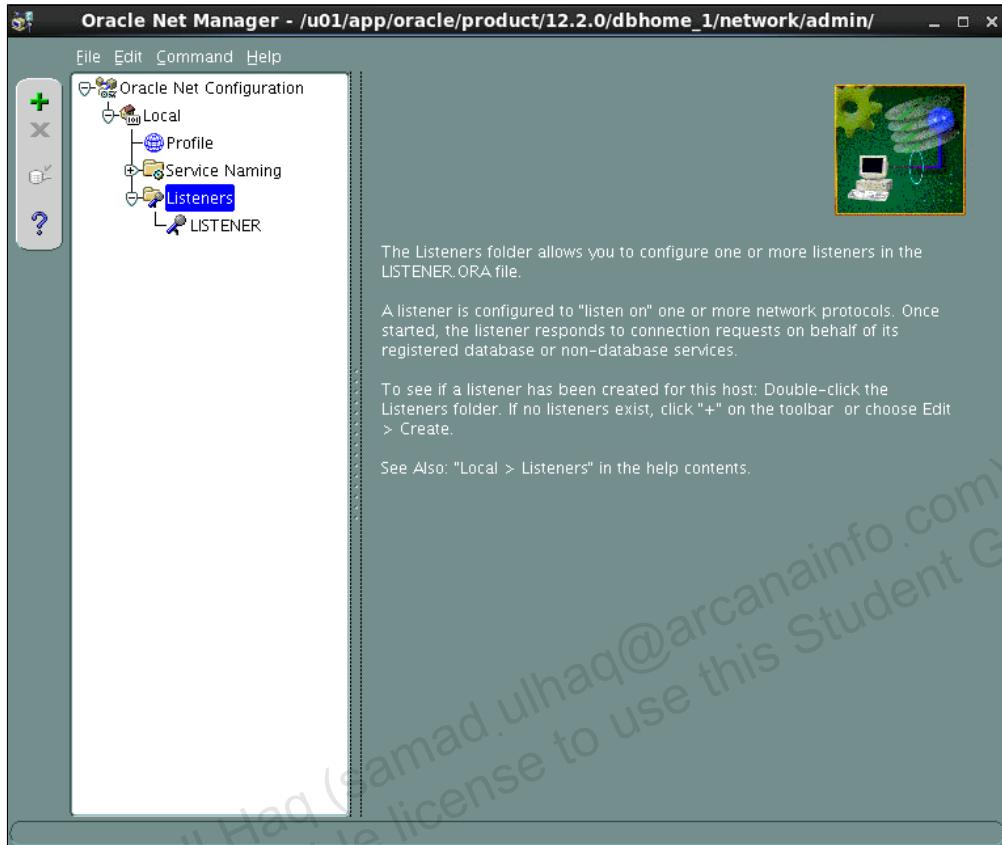
```
LISTENER2 =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1561))
)
ADR_BASE_LISTENER2 = /u01/app/oracle
```

- e. Save the file, close it, and then rename it back to `listener.ora`.
2. Complete this step if you want to use Oracle Net Manager to configure the `listener.ora` file. Do not perform this step if you performed step 1.

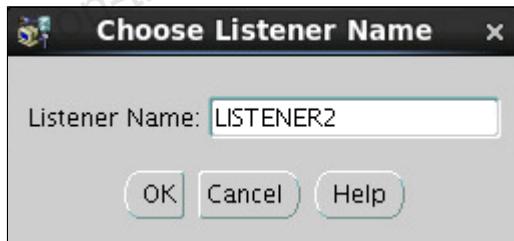
- a. In the terminal window, start Oracle Net Manager.

```
$ netmgr
```

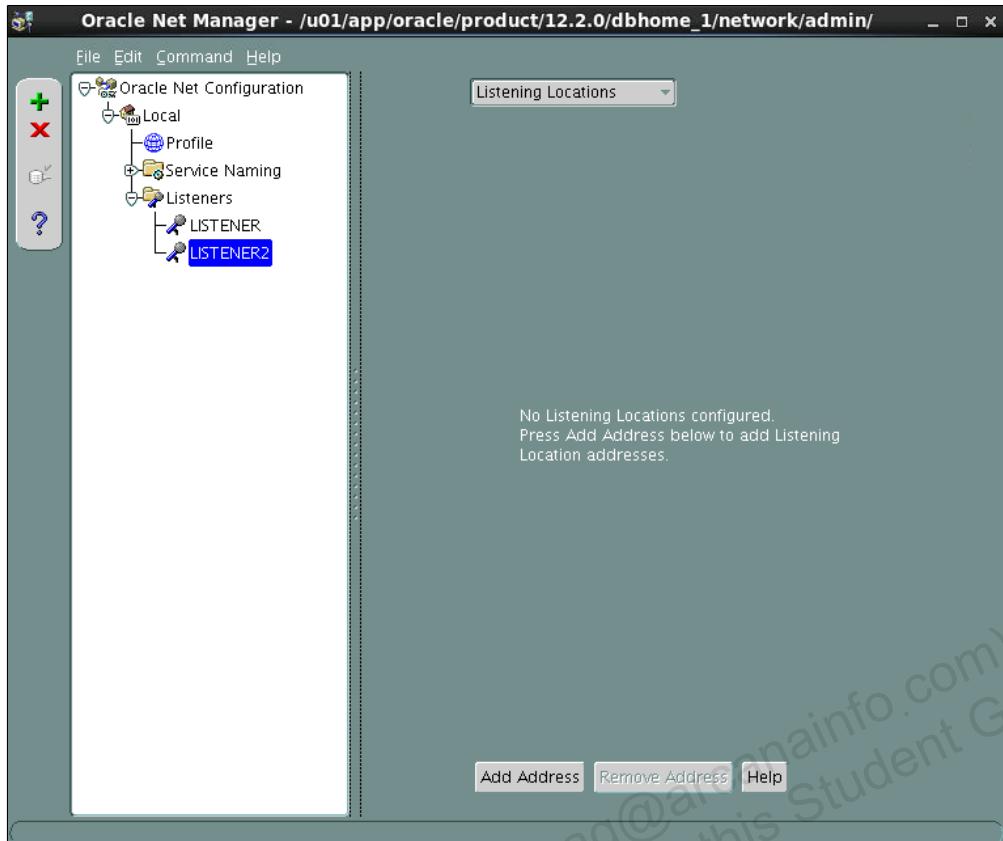
- b. In the Oracle Net Manager navigation pane, expand **Local**, and then **Listeners**. The default listener, LISTENER, is listed. You created this listener when you created the CDB and PDB in Lesson 1. Select **Listeners** and click the green plus sign to begin defining a new listener.



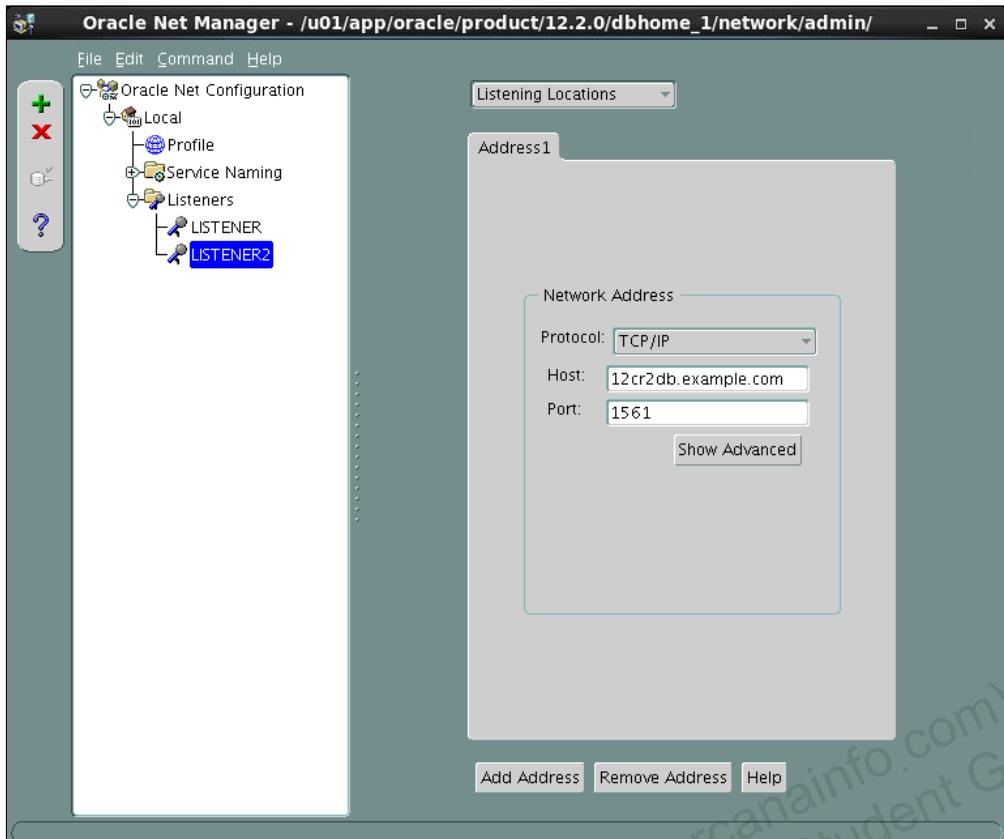
- c. In the Choose Listener Name dialog box, enter **LISTENER2**, and click **OK**. LISTENER2 is added to the list of listeners.



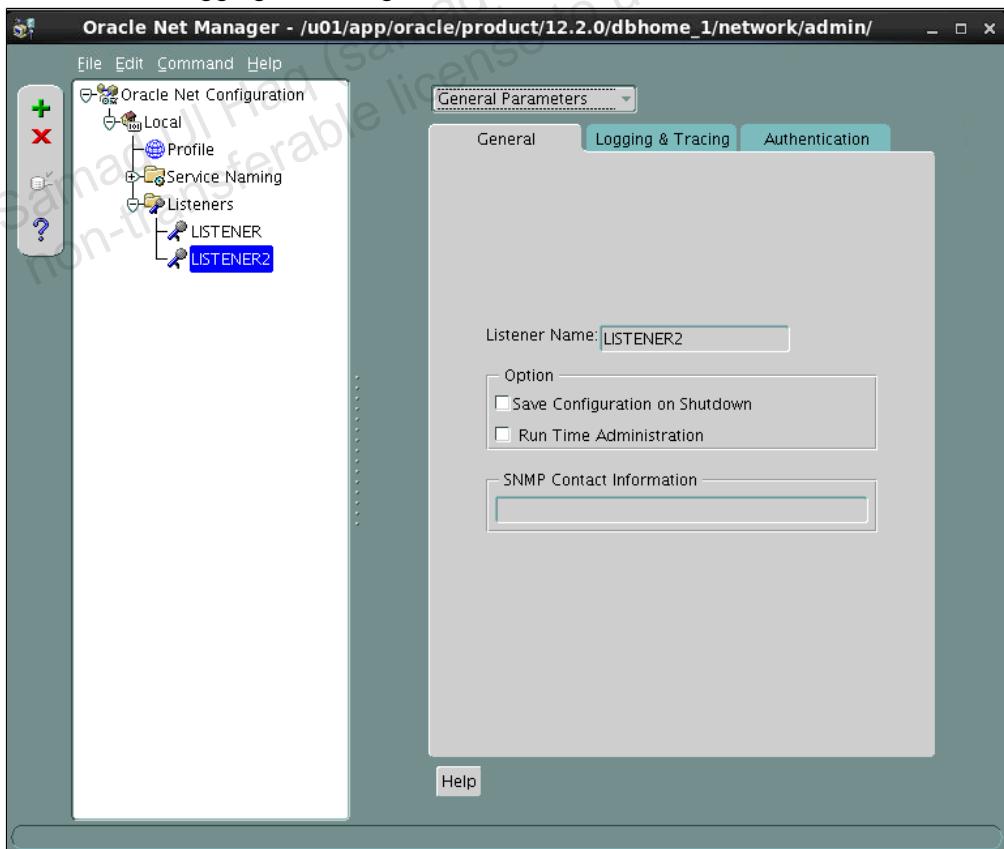
- d. With LISTENER2 selected on the left side, click **Add Address** on the right side.



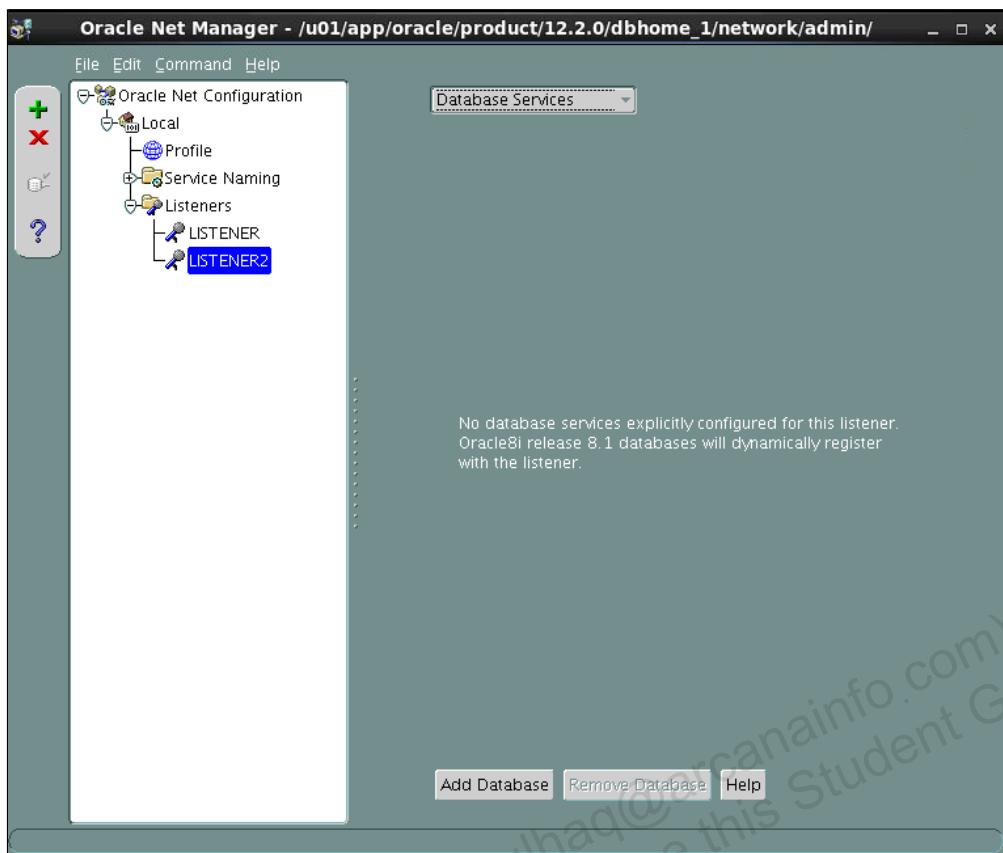
- e. In the Address1 panel on the right, leave **Listening Locations** selected in the drop-down list, leave **TCP/IP** selected as the protocol, set the host name to **12cr2db.example.com**. In the Port box, enter **1561**.



- f. For interest, select **General Parameters** from the drop-down list. Review the configuration options on the General, Logging & Tracing, and Authentication tabs.



- g. Select **Database Services** from the drop-down list. There are no databases currently configured for the listener.



- h. Select **File** and then **Save Network Configuration**.
- i. Select **File**, and then **Exit** to exit Oracle Net Manager. You just added an entry into the `listeners.ora` file.
- j. On your desktop, open **Computer**, and then **Filesystem**.
- k. Browse to `$ORACLE_HOME/network/admin` (`/u01/app/oracle/product/12.2.0/dbhome_1/network/admin`).
- l. Change the name of the `listener.ora` file to `listener.ora.txt`, and then open it in gedit. Notice that 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.
Note: Your domain name for `LISTENER` is different than the one shown below.

```

# listener.ora Network Configuration File:
/u01/app/oracle/product/12.2.0/dbhome_1/network/admin/listener.ora
# Generated by Oracle configuration tools.

LISTENER2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1561))
  )

ADR_BASE_LISTENER2 = /u01/app/oracle

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))
    )
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )
)

ADR_BASE_LISTENER = /u01/app/oracle

```

- m. Close the file, and then rename it back to `listener.ora`.
- n. Close the "admin - File Browser" window.

Check the Status of LISTENER2

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

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

```

$ lsnrctl

LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 19-AUG-2016 20:39:49
Copyright (c) 1991, 2016, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL>

```

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

```
LSNRCTL> status LISTENER2
```

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST= 12cr2db.example.com )(PORT=1561)))
TNS-12541: TNS:no listener
TNS-12560: TNS:protocol adapter error
TNS-00511: No listener
Linux Error: 111: Connection refused
LSNRCTL>
```

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

```
LSNRCTL> start LISTENER2
```

```
Starting /u01/app/oracle/product/12.2.0/dbhome_1/bin/tnslnsr: please wait...
```

```
TNSLSNR for Linux: Version 12.2.0.1.0 - Production
System parameter file is
/u01/app/oracle/product/12.2.0/dbhome_1/network/admin/listener.ora
Log messages written to
/u01/app/oracle/diag/tnslnsr/12cr2db/listener2/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST= 12cr2db.example.com )(PORT=1561)))
```

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST= 12cr2db.example.com )(PORT=1561)))
STATUS of the LISTENER
```

```
-----
Alias                      LISTENER2
Version                    TNSLSNR for Linux: Version 12.2.0.1.0 - Production
Start Date                 19-AUG-2016 22:02:37
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/12.2.0/dbhome_1/network/admin/listener.ora
Listener Log File          /u01/app/oracle/diag/tnslnsr/12cr2db/listener2/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST= 12cr2db.example.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.
- Wait about 60 seconds, and then check the status of LISTENER2 again. By then, the LREG process will have time to register the database services with your listener.

```

LSNRCTL> status LISTENER2

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST= 12cr2db.example.com
)(PORT=1561)))
STATUS of the LISTENER
-----
Alias                      LISTENER2
Version                    TNSLSNR for Linux: Version 12.2.0.1.0 - Production
Start Date                 19-AUG-2016 22:02:37
Uptime                     0 days 0 hr. 2 min. 27 sec
Trace Level                off
Security                   ON: Local OS Authentication
SNMP                       OFF

Listener Parameter File    /u01/app/oracle/product/12.2.0/dbhome_1/network/admin/listener.ora
Listener Log File          /u01/app/oracle/diag/tnslsnr/12cr2db/listener2/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST= 12cr2db.example.com )(PORT=1561)))
Services Summary...
Service " 379d6befb4634ac8e0530d11ed0ad32d.example.com " has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service " 38046a0b01822d55e0530d11ed0afe33.example.com " has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service " ORCL.example.com " has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service " ORCLXDB.example.com " has 1 instance(s).
  Instance "ORCL", status READY, has 1 handler(s) for this service...
Service " pdb1.example.com " has 1 instance(s). Instance "ORCL", status READY,
has 1 handler(s) for this service... Service " pdb2.example.com " has 1
instance(s). Instance "ORCL", status READY, has 1 handler(s) for this service...
The command completed successfully
LSNRCTL>

```

6. Exit the Listener Control utility.

```

LSNRCTL> EXIT

$
```

Connect to a Database Service Through LISTENER2

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

1. Using Easy Connect syntax, start SQL*Plus and connect to the CDB using LISTENER2. Make sure to specify the non-default port number **1561**. See [Appendix - Product-Specific Credentials](#) for the password. The connection succeeds.

```
$ sqlplus system/<password>@localhost:1561/ ORCL.example.com

SQL*Plus: Release 12.2.0.1.0 Production on Fri Aug 19 22:08:53
2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Last Successful login time: Fri Aug 19 2016 21:09:25 +00:00
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

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

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

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 4-3 Creating a Static Listener for a PDB

Overview

In this practice, you create a listener named LISTENER_PDB1 that listens on the non-default port 1562 for the PDB1.example.com service. Configure the listener to use static service registration.

Assumptions

You are logged in as the oracle user on VM1.

Tasks

Add an Entry to the listener.ora File

1. On your desktop, open **Computer**, and then **Filesystem**.
2. Browse to **\$ORACLE_HOME/network/admin** (`/u01/app/oracle/product/12.2.0/dbhome_1/network/admin`).
3. Change the name of the `listener.ora` file to `listener.ora.txt`, and then open it in gedit.
4. Add the following entry above the first entry, but below the two comments at the top. Examine the `SID_LIST_LISTENER_PDB1` section. Include the PDB service name in the `GLOBAL_DBNAME` parameter, the database instance name in the `SID_NAME` parameter, and the Oracle home directory in the `ORACLE_HOME` parameter. Enter this information manually, and avoid using copy and paste.

```
LISTENER_PDB1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1562))
  )
SID_LIST_LISTENER_PDB1 =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = PDB1.example.com)
      (SID_NAME = ORCL)
      (ORACLE_HOME = /u01/app/oracle/product/12.2.0/dbhome_1)
    )
  )
```

5. Save the file, close it, and exit gedit.
6. Rename the `listener.ora.txt` file back to `listener.ora`.
7. Close the "admin - File Browser" window.

Start LISTENER_PDB1

Using the Listener Control utility, start LISTENER_PDB1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

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

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

```
$ lsnrctl  
  
LSNRCTL for Linux: Version 12.2.0.1.0 - Production on 19-AUG-2016 20:39:49  
Copyright (c) 1991, 2016, Oracle. All rights reserved.  
Welcome to LSNRCTL, type "help" for information.  
LSNRCTL>
```

3. Check the status of LISTENER_PDB1 by issuing the **status** command. Your results indicate "no listener" and "Connection refused" because you just created the listener and need to start it.

```
LSNRCTL> status LISTENER_PDB1  
  
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST= 12cr2db.example.com  
(PORT=1562)))  
TNS-12541: TNS:no listener  
TNS-12560: TNS:protocol adapter error  
TNS-00511: No listener  
Linux Error: 111: Connection refused  
LSNRCTL>
```

4. Start LISTENER_PDB1 by issuing the **start LISTENER_PDB1** command. The results show that the listener has one service registered with it: PDB1.example.com. The status of the service is unknown. This is the case for static listeners. The LREG service doesn't update a static listener with database instance information.

Samad Ullah
non-transferable
use this Student Guide
arcana@samadullahq@arcanaimc.com has a

```

LSNRCTL> start LISTENER_PDB1

Starting /u01/app/oracle/product/12.2.0/dbhome_1/bin/tnslnsr: please wait...

TNSLSNR for Linux: Version 12.2.0.1.0 - Production
System parameter file is
/u01/app/oracle/product/12.2.0/dbhome_1/network/admin/listener.ora
Log messages written to
/u01/app/oracle/diag/tnslnsr/12cr2db/listener_pdb1/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST= 12cr2db.example.com )(PORT=1562)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST= 12cr2db.example.com )(PORT=1562)))
STATUS of the LISTENER
-----
Alias                      LISTENER_PDB1
Version                    TNSLSNR for Linux: Version 12.2.0.1.0 - Production
Start Date                 19-AUG-2016 22:45:18
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/12.2.0/dbhome_1/network/admin/listener.ora
Listener Log File          /u01/app/oracle/diag/tnslnsr/12cr2db/listener_pdb1/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST= 12cr2db.example.com )(PORT=1562)))
Services Summary...
Service " PDB1.example.com " has 1 instance(s).
  Instance " ORCL ", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
LSNRCTL>

```

5. Exit the Listener Control utility.

```
LSNRCTL> EXIT
```

```
$
```

Connect to a Database Service Through LISTENER_PDB1

Test the connection to the PDB1.example.com service through the LISTENER_PDB1 listener.

1. Using Easy Connect syntax, start SQL*Plus and connect to PDB1 using LISTENER_PDB1. Make sure to specify the non-default port number 1562. See [Appendix - Product-Specific Credentials](#) for the password. The connection succeeds.

```
$ sqlplus system/<password>@localhost:1562/PDB1.example.com

SQL*Plus: Release 12.2.0.1.0 Production on Fri Aug 19 22:08:53
2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
Last Successful login time: Fri Aug 19 2016 21:09:25 +00:00
Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

2. Show the current container name. The result is PDB1. You have successfully configured a static listener to support the PDB1.example.com service.

```
SQL> SHOW con_name

CON_NAME
-----
PDB1
SQL>
```

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

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

Practice 4-4 Creating a Net Service Name

Overview

In this practice, you create a net service name called MyPDB1 by using a text editor or Oracle Net Configuration Assistant (netca).

Remember, you can also use the Oracle Net Manager (netmgr) utility to create net service names.

Assumptions

You are logged in to VM1 as the oracle user.

Tasks

Create a Net Service Named Called MyPDB1

1. Open a new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL .

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Locate and view your local tnsnames.ora file before you add a net service name to it.

- a. Change the directory to /u01/app/oracle/product/12.2.0/dbhome_1/network/admin. This directory can also be specified as \$ORACLE_HOME/network/admin.

Tip: You can search for the tnsnames.ora file by starting the /u01 directory and issuing the following command: find . -name tnsnames.ora.

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

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

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

- c. View the tnsnames.ora file by using the cat command. When your CDB and PDB were created, DBCA had automatically created a net service name called ORCL, which accesses the entire CDB. Later, the PDB1 and PDB2 net service names were added to VM1 by the developer to make it fast and easy for you to connect to PDB1 and PDB2.

Let's examine the ORCL entry. The connect descriptor encompasses all the information below ORCL, starting with (DESCRIPTION = ... to (SERVICE_NAME = ORCL.example.com). The ADDRESS section

describes the listener. The CONNECT_DATA section describes the database service. What you want to do next is add another entry to this file that specifies the connectivity information for PDB1. Your net service name will be similar to the PDB1 entry. Note: The LISTENER2 entry shown below was created in [Practice 4-2 Creating a Dynamic Listener](#). You do not require LISTENER2 for this practice.

```
$ cat tnsnames.ora

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

LISTENER2 =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com )(PORT = 1561))

LISTENER_ORCL =
  (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com )(PORT = 1521))

PDB1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com )(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PDB1.example.com)
    )
  )

PDB2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com )(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PDB2.example.com)
    )
  )

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com )(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL.example.com )
    )
  )
$
```

3. To create a net service name by using the vi editor:

- Open `tnsnames.ora` and

Note: If you want to use netca to create the net service name, skip this step and complete step 4.

```
$ vi tnsnames.ora
```

- b. Insert the following net service name entry called MYPDB1 after the PDB1 entry. Enter this information manually, and avoid using copy and paste. Save the file and quit the vi editor (:wq).

```
MYPDB1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = PDB1.example.com )
    )
  )
```

4. To create a net service name for PDB1 with netca, complete the following steps. Skip this step if you already created the net service name in step 3.

- a. In your terminal window, start **netca**.

```
$ netca
```

- b. On the Welcome page, select **Local Net Service Name configuration**, and click **Next**.



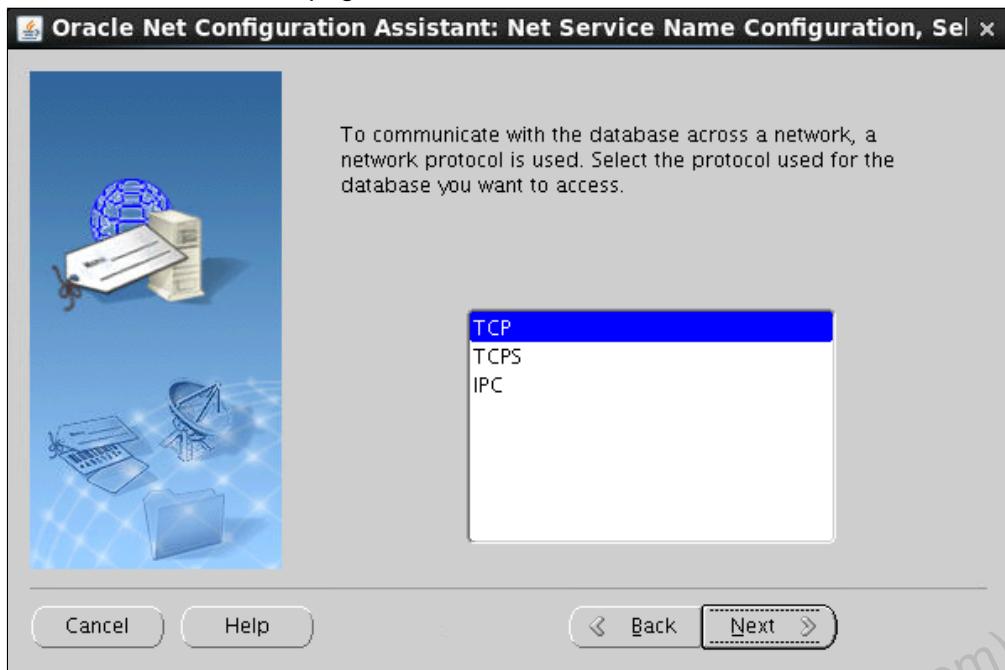
- c. On the Net Service Name Configuration page, leave **Add** selected, and click **Next**.



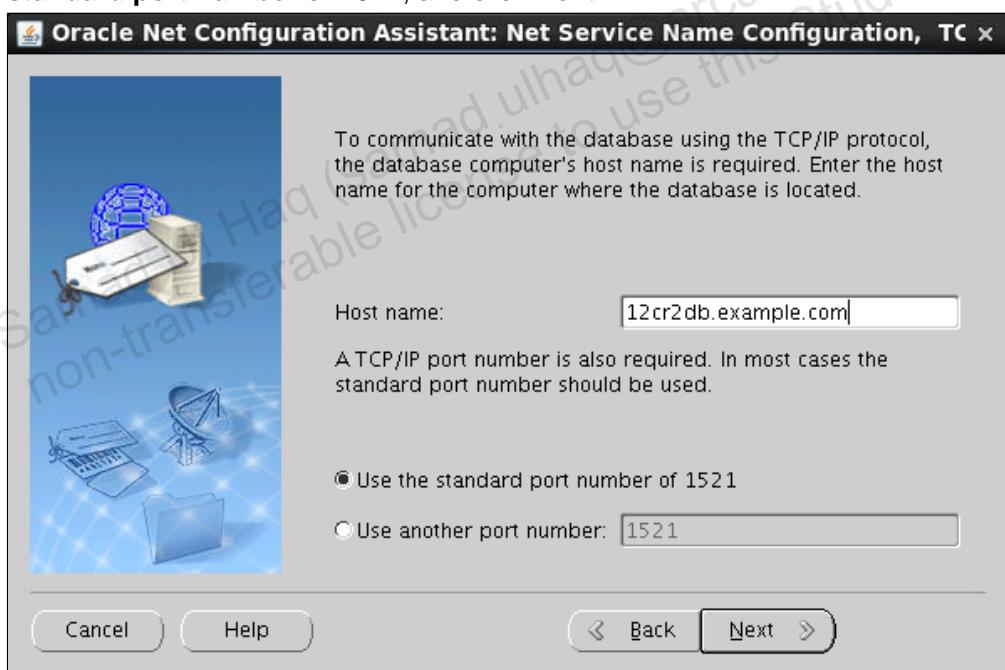
- d. On the Service Name page, enter the full database service name, **PDB1.example.com** (case is not important), and click **Next**.



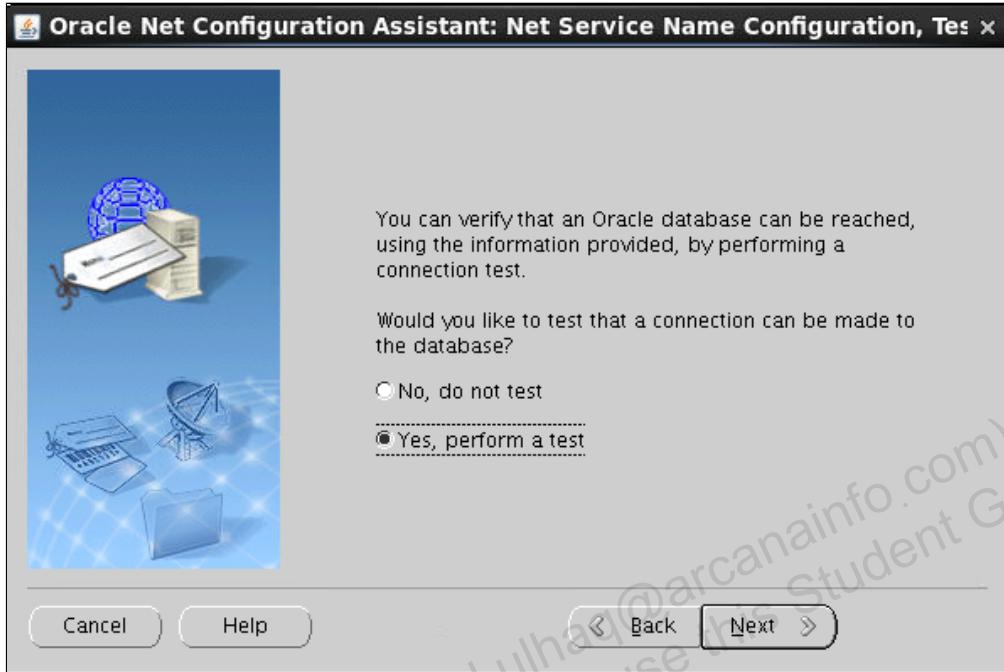
- e. On the Select Protocols page, select **TCP** and click **Next**.



- f. On the TCP/IP Protocol page, enter **12cr2db.example.com** for the host name, select **Use the standard port number of 1521**, and click **Next**.



- g. On the Test page, select **Yes, perform a test**, and click **Next**.



- h. On the Login page, click **Change Login**.

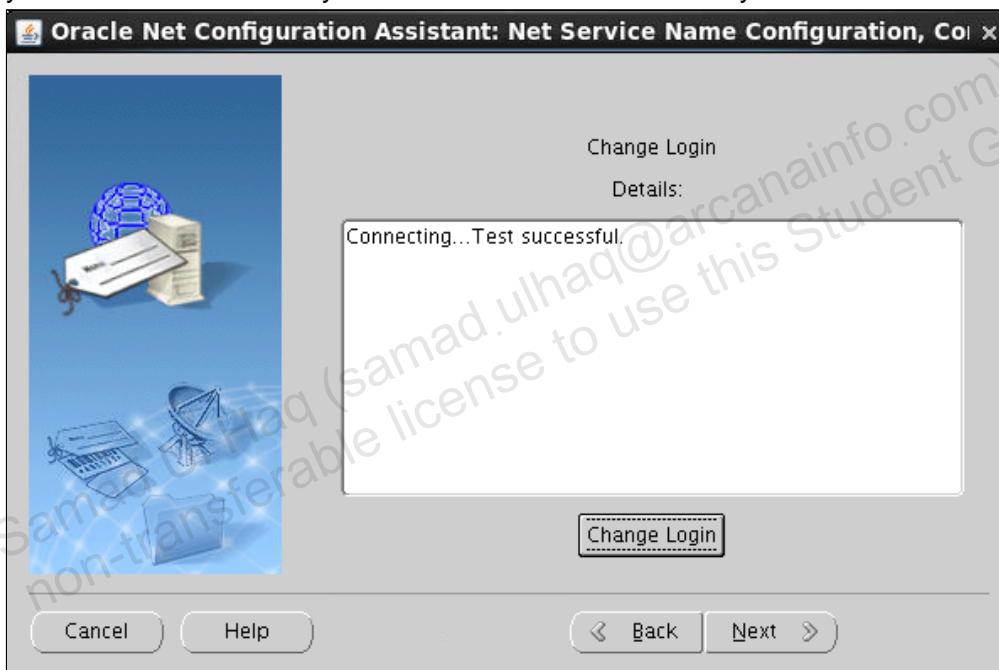


- i. In the Change Login dialog box, enter the username **PDBADMIN**, the password, and click **OK**. See Appendix - Product-Specific Credentials for the password.

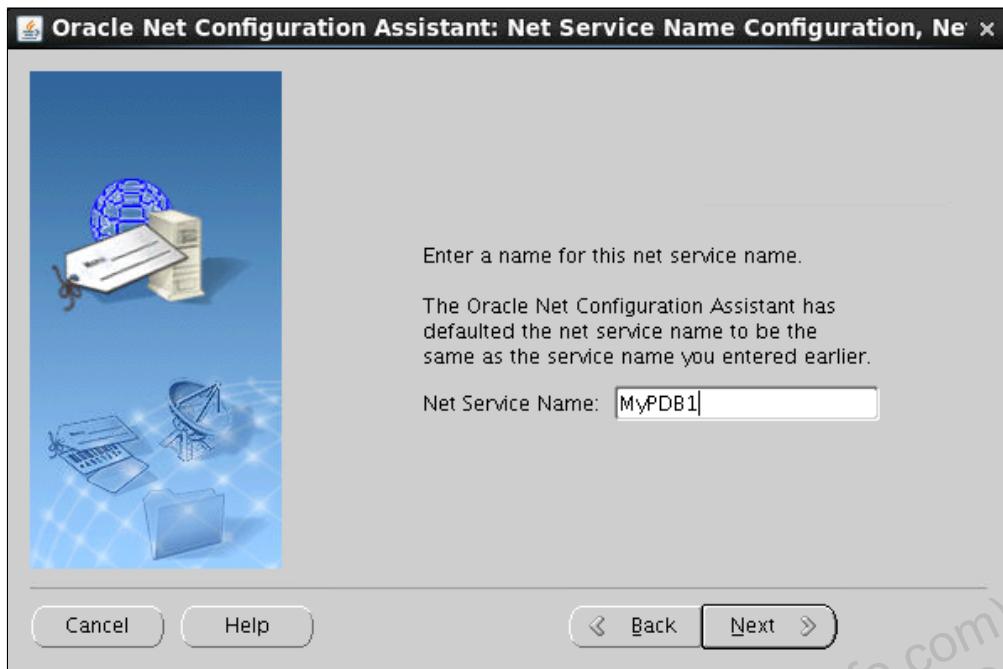
PDBADMIN was created by DBCA when DBCA created PDB1. This user is specifically meant to be an administrative user for PDB1. You can also log in as the SYSTEM user to do the test.



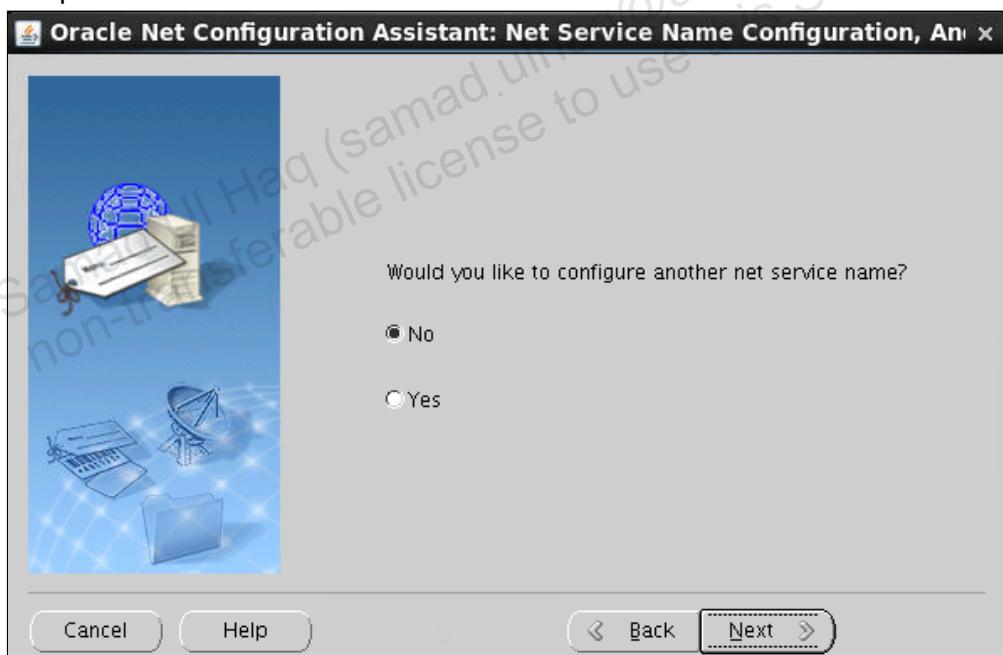
- j. If the test is successful (the details pane will tell you so), click **Next**. Otherwise, click **Back** as much as you need and make sure you've entered the details correctly.



- k. On the Net Service Name page, enter **MyPDB1** as the net service name, and click **Next**. Note: Entering MyPDB1 in netca results in MYPDB1 (all capitals) in the `tnsnames.ora` file.



- l. On the Another Net Service Name page, leave **No** selected, and click **Next**. The configuration is completed.



- m. On the Done page, click **Next**. You're returned to the Welcome page.



- n. Click **Finish** to close netca.



- o. In the terminal window, notice the message that says the net service name, PDB1, was successfully created.

```
Oracle Net Services Configuration:  
Default local naming configuration complete.  
Created net service name: MyPDB1  
Oracle Net Services configuration successful. The exit code is 0
```

- p. Change the directory to /u01/app/oracle/product/12.2.0/dbhome_1/network/admin.

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

- q. Open the tnsnames.ora file by using the cat command, and verify that the MYPDB1 net service name entry is listed.

```
$ cat tnsnames.ora  
  
...  
MYPDB1 =  
(DESCRIPTION =  
(ADDRESS_LIST =  
(ADDRESS = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))  
)  
(CONNECT_DATA =  
(SERVICE_NAME = PDB1.example.com)  
)  
)  
...  
$
```

5. Test the Oracle Net service alias by using the tnsping utility. The last line in the results indicates that the connection is OK, which tells you that there is connectivity between the client and the Oracle Net Listener. It does not tell you whether the requested service (PDB1.example.com) is available.

```
$ tnsping MyPDB1
```

TNS Ping Utility for Linux: Version 12.2.0.1.0 - Production on 29-AUG-2016
18:15:57

Copyright (c) 1997, 2016, Oracle. All rights reserved.

Used parameter files:

/u01/app/oracle/product/12.2.0/dbhome_1/network/admin/sqlnet.ora

Used TNSNAMES adapter to resolve the alias

Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (PROTOCOL = TCP)(HOST = 12cr2db.example.com)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME = PDB1.example.com))

OK (0 msec)

\$

6. Connect to PDB1, verify the current container, and then exit SQL*Plus.

- Start SQL*Plus and connect to PDB1 as the system user by using the MyPDB1 net service name. See [Appendix - Product-Specific Credentials](#) for the password.

Recall that in an earlier practice you needed to use the EasyConnect syntax to make a direct connection to a PDB by using the SQL statement

`CONNECT system/<password>@12cr2db.example.com:1521/PDB1.example.com`. Using a net service name, like the one you just created, is much simpler.

```
$ sqlplus system/<password>@MyPDB1
```

SQL*Plus: Release 12.2.0.1.2 Production on Sat Aug 20 02:21:14 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:

Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
SQL>

- Verify that the current container name is PDB1.

```
SQL> SHOW con_name
```

CON_NAME

PDB1
SQL>

PDB1
SQL>

- c. Exit from SQL*Plus and close the terminal window.

```
SQL> EXIT  
.  
$
```

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 4-5 Creating a Database Link to an External Database

Overview

In this practice, you use Oracle Net Configuration Assistant (netca) to configure a net service name for an external Oracle database located on VM2. Then, from the Oracle database on VM1, you create a database link to the Oracle database on VM2.

Before you create the net service name, connect to VM2 and discover the full database server name. The database service name, along with the full host server name for VM2 (which you're given), is the information you need to create a net service name.

Assumptions

You are logged in to VM1 as the `oracle` user.

You have a second virtual machine, VM2, in your training environment configured with a non-CDB. This Oracle database is installed to support Enterprise Manager Cloud Control.

The full host server name for VM2 is `em13c.example.com`.

Tasks

Find the Full Database Service Name on VM2

1. Start a new TigerVNC session and connect to VM2 by entering the full VM name, `em13c.example.com`. When prompted for the operating system login, enter the password. Your instructor will provide you with this password. This TigerVNC session will be referred to as session 2. The TigerVNC session for VM1 will be referred to as session 1.
2. Open a terminal window and issue the `more /etc/oratab` command. The first word in the last line of the results tells you the full database service name, which is `em13rep`.

```
$ more /etc/oratab
#
# This file is used by ORACLE utilities. It is created by root.sh
# and updated by either Database Configuration Assistant while creating
# a database or ASM Configuration Assistant while creating ASM instance.

# A colon, ':', is used as the field terminator. A new line terminates
# the entry. Lines beginning with a pound sign, '#', are comments.
#
# Entries are of the form:
#   $ORACLE_SID:$ORACLE_HOME:<N|Y>:
#
# The first and second fields are the system identifier and home
# directory of the database respectively. The third field indicates
# to the dbstart utility that the database should , "Y", or should not,
# "N", be brought up at system boot time.
#
# Multiple entries with the same $ORACLE_SID are not allowed.
#
#
em13rep:/u01/app/oracle/product/12.1.0/dbhome_1:Y
```

3. Another way to find the database service name is to examine the directory structure for the Oracle database installation.
 - a. Minimize the terminal window.
 - b. On your desktop, open **Computer** and then **Filesystem**. Browse to `/u01/app/oracle/oradata`. Notice that there is a directory named `em13rep`. This directory name tells you the name of the SID, which is *usually* the same name as the database service name. Close the window and return to the terminal window.
4. Open a new terminal window and source the `oraenv` script to set the Oracle environment variables. When prompted, enter **em13rep** for the `ORACLE_SID` value. Case is important.

```
$ . oraenv
ORACLE_SID = [oracle] ? em13rep

The Oracle base has been set to /u01/app/oracle
$
```

5. Start SQL*Plus and log in to the database with the `SYSDBA` privilege. You must use lowercase letters for `sqlplus`.
You are logged in as the `SYS` user. The database is version 12.1.0.2.0.

```
$ sqlplus / as sysdba
```

```
SQL*Plus: Release 12.1.0.2.0 Production on Tue Apr 26 19:48:38 2016
Copyright (c) 1982, 2014, Oracle. All rights reserved.
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing
options
SQL>
```

6. Find out whether the Oracle database is a non-CDB or a CDB by querying the V\$DATABASE view. The query results indicate that the database is a non-CDB. This fact will not impact how you create a net service name to the database.

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

NAME      CDB      CON_ID
-----  -----
EM13REP    NO        0
SQL>
```

7. List all the services for the database by querying the V\$SERVICES view. The query should return four services: em13repXDB, SYS\$USERS, SYS\$BACKGROUND, and em13rep. The service named em13rep is the name that you will enter as the database service name in netca when you create a net service name later in the practice.

```
SQL> SELECT con_id, name FROM v$services order by 1;

CON_ID NAME
----- 
0 em13repXDB
0 SYS$USERS
0 SYS$BACKGROUND
0 em13rep
SQL>
```

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

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

9. Close your TigerVNC connection to VM2 (session 2).

Create the Net Service Name em13rep on VM1

1. Return to your TigerVNC connection to VM1 (session 1).
2. Open a new terminal window and set the Oracle environment variables to your local ORCL database.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL
```

```
The Oracle base has been set to /u01/app/oracle
$
```

3. Create a net service name called EM13REP by using the vi editor. Alternatively, skip this step and follow step 4 to create the net service name by using netca.

- a. Change the directory to /u01/app/oracle/product/12.2.0/dbhome_1/network/admin.

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

- b. Open tnsnames.ora in the vi editor.

```
$ vi tnsnames.ora
```

- c. Add the following entry for EM13REP. Enter this information manually, and avoid using copy and paste.

```
EM13REP =
(DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = em13c.example.com)(PORT = 1521))
  )
  (CONNECT_DATA =
    (SERVICE_NAME = em13rep)
  )
)
```

- d. Save the file and quit the vi editor (:wq).

4. To create the net service name by using netca, perform the following steps. If you completed step 3, you can skip this step.

- a. Start netca.

```
$ netca
```

- b. On the Welcome page, select **Local Net Service Name configuration**, and click **Next**.



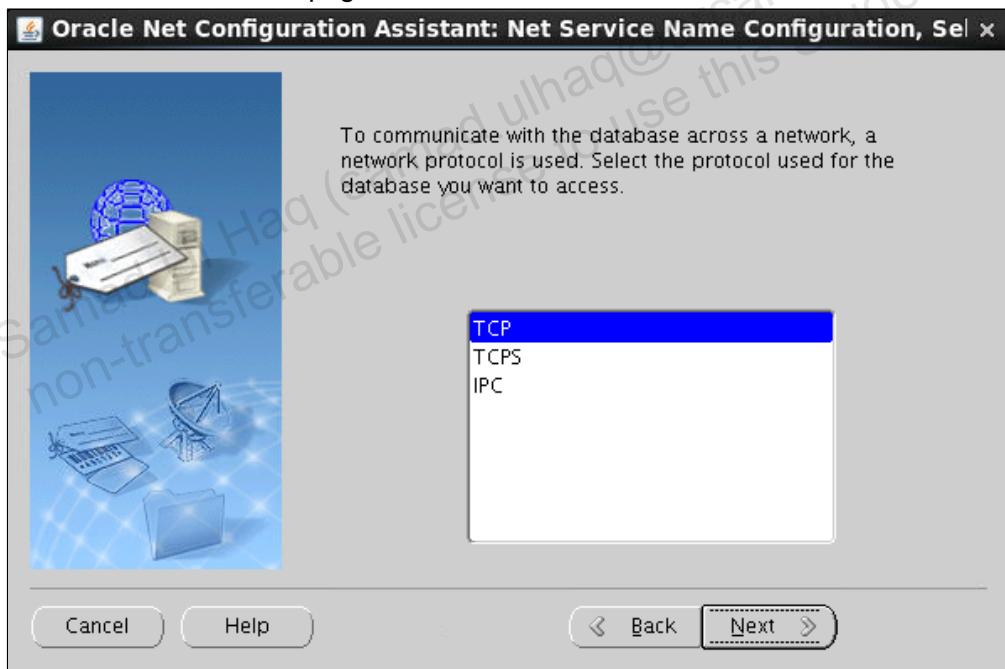
- c. On the Net Service Name Configuration page, leave **Add** selected, and click **Next**.



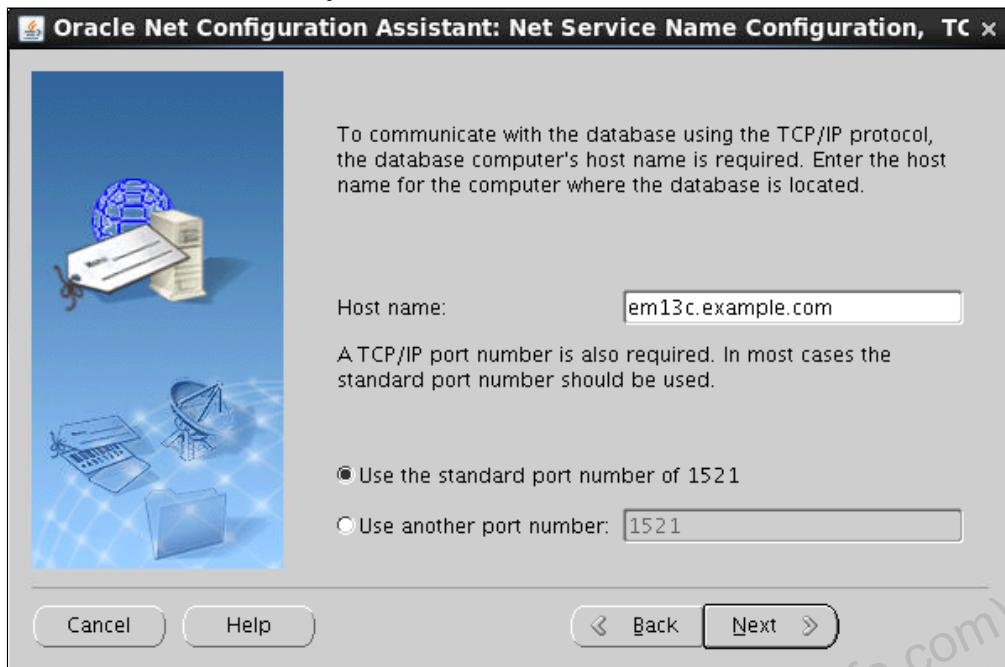
- d. On the Service Name page, enter the full database service name, **em13rep** (case is important), and click **Next**.



- e. On the Select Protocols page, leave **TCP** selected, and click **Next**.



- f. On the TCP/IP Protocol page, enter the full server host name for VM2, which is `em13c.example.com`. Leave **Use the standard port number of 1521** selected, and click **Next**.



- g. On the Test page, select **Yes, perform a test**, and click **Next**.



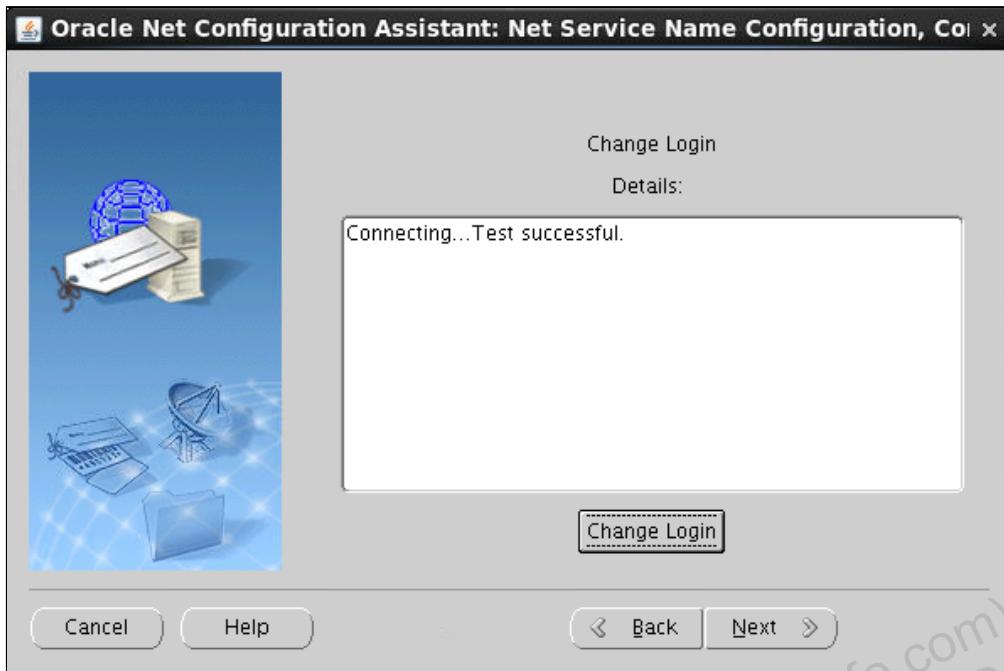
- h. On the Login page, click **Change Login**.



- i. In the Change Login dialog box, enter the username **SYSTEM**, the password, and click **OK**. See Appendix - Product-Specific Credentials for the password.



- j. If the test is successful (the Details pane will tell you so), click **Next**. Otherwise, click **Back** as much as you need and make sure you've entered the details correctly.



- k. On the Net Service Name page, leave **em13rep** as the net service name, and click **Next**.

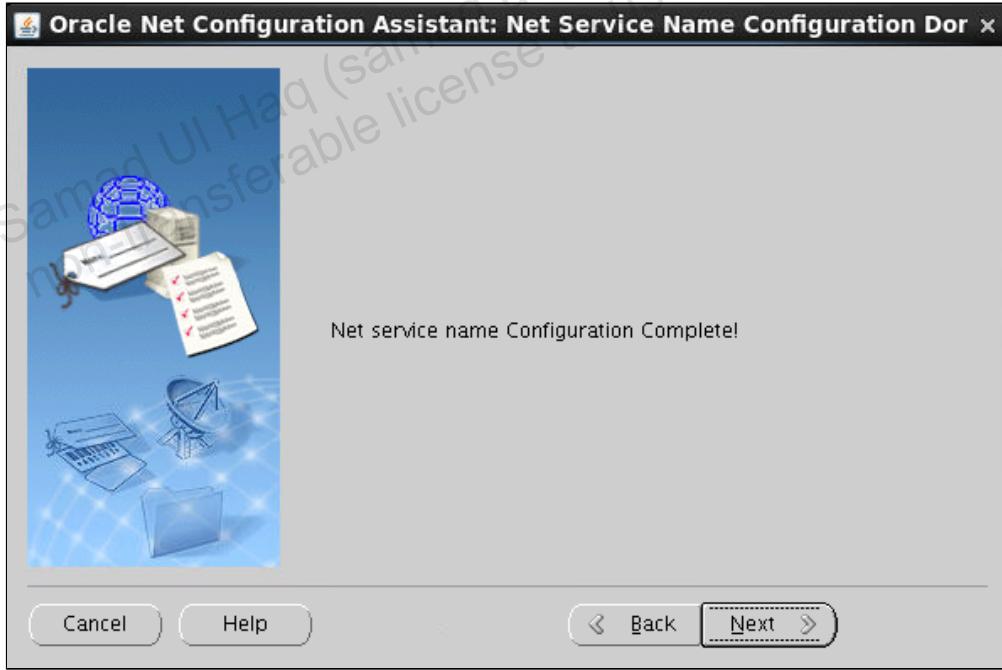
Note: Here you can enter any net service name you want. The netca utility defaults the net service name to be the same as the database service name you entered earlier.



- l. On the Another Net Service Name page, leave **No** selected, and click **Next**. The configuration is completed.



- m. On the Done page, click **Next**. You'll be returned to the Welcome page.



- n. Click **Finish** to close netca.



- o. In the terminal window, verify that the message says that the net service name, em13rep, was successfully created.

```
Oracle Net Services Configuration:  
Default local naming configuration complete.  
Created net service name: em13rep  
Oracle Net Services configuration successful. The exit code is 0
```

- o.
- p. Change the directory to /u01/app/oracle/product/12.2.0/dbhome_1/network/admin.

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

- q. View the tnsnames.ora file by using the cat command and verify the entry for EM13REP is as follows:

```
$ cat tnsnames.ora

...
EM13REP =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = em13c.example.com)(PORT = 1521))

    )
    (CONNECT_DATA =
      (SERVICE_NAME = em13rep)
    )
  )
...
$
```

Test the Net Service Name from VM1

1. On VM1, connect to the Oracle database on VM2 as the SYSTEM user by using the net service name you just created. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@em13rep

SQL*Plus: Release 12.2.0.1.2 Production on Sat Aug 20 02:21:14 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

2. Verify that the current container name is em13rep. The results confirm that you are successfully connected to the external database.

```
SQL> SHOW con_name

CON_NAME
-----
em13rep
SQL>
```

3. Disconnect from the external database.

```
SQL> DISCONNECT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 -
64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing
options
SQL>
```

Create a Database Link to the External Database

1. In SQL*Plus, connect to the root container of your database on VM1.

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
```

2. Verify that the current container name is CDB\$ROOT.

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

3. Create a fixed user database link named mylink that connects to the SYSTEM user on the external database (on VM2). See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CREATE DATABASE LINK mylink CONNECT TO SYSTEM IDENTIFIED BY <password>
USING 'em13rep';
Database link created.
SQL>
```

4. View the description of the view named dba_db_links. This view describes all database links in the database.

```
SQL> DESC dba_db_links
Name                           Null?    Type
-----                         -----
OWNER                          NOT NULL VARCHAR2(128)
DB_LINK                        NOT NULL VARCHAR2(128)
USERNAME                       VARCHAR2(128)
HOST                           VARCHAR2(2000)
CREATED                        NOT NULL DATE
HIDDEN                         VARCHAR2(3)
SQL>
```

5. Query the dba_db_links view. Your database link is listed.

```
SQL> COLUMN owner FORMAT A10
SQL> COLUMN db_link FORMAT A20
SQL> COLUMN username FORMAT A10
SQL> SELECT owner, db_link, username FROM dba_db_links;
OWNER      DB_LINK          USERNAME
-----      -----
SYS        SYS_HUB
SYS        MYLINK.EXAMPLE.COM   SYSTEM
SQL>
```

6. Query the V\$DATABASE view in the external database by using the database link. The name column shows the EM13REP container name and the CDB column indicates that the database is a non-container database. You have successfully queried the external database using a database link.

```
SQL> SELECT name, cdb FROM v$database@mylink;  
NAME          CDB  
-----  -----  
EM13REP      NO  
SQL>
```

7. Exit from SQL*Plus and close the terminal window.

```
SQL> EXIT  
:  
$
```

5

Administering User Security



ORACLE®

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Objectives for Lesson 5

Unauthorized reproduction or distribution prohibited
Copyright © 2017, Oracle and/or its affiliates.

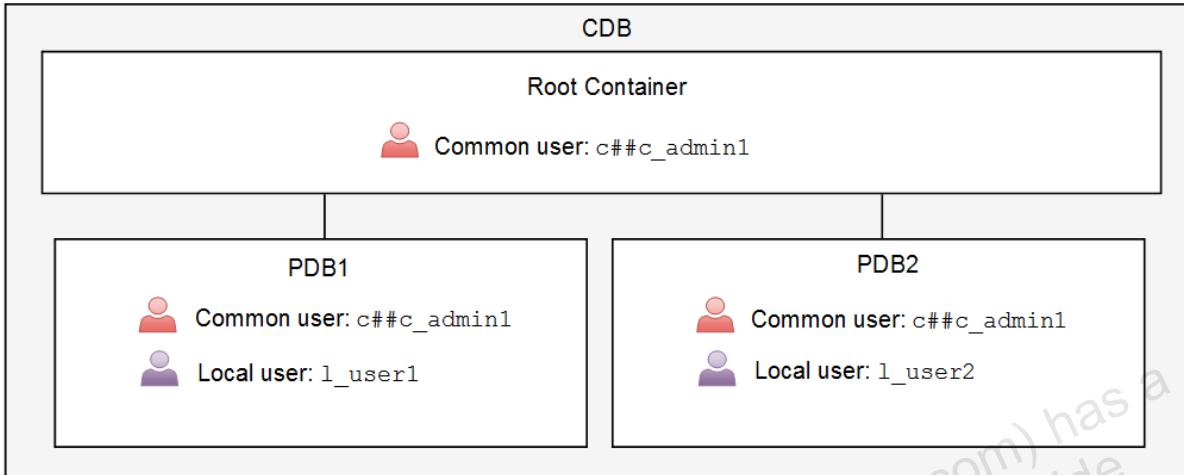
After completing this lesson, you should be able to:

- Create database users
- Grant privileges to database users
- Create and grant roles to users or other roles
- Revoke privileges and roles from users and other roles
- Create and assign profiles to users
- Explain the various authentication options for users
- Assign quota to users
- Apply the principal of least privilege



ORACLE®

Creating Users



ORACLE

Common Versus Local Users

A multitenant environment has two types of database users that you can create:

- **Common user:** The user is replicated in all existing and future containers. Oracle supplies several common user accounts for database administrators to use. A common user that you create, by default, must be given a name that starts with C## or c##, for example, c##c_admin1. The COMMON_USER_PREFIX parameter specifies a prefix for common users, roles, and profiles in a container database. To create a common user, log in to the root container, issue the CREATE USER command and include the CONTAINER=ALL clause, for example:

```
SQL> CONNECT / AS SYSDBA
SQL> CREATE USER c##c_admin1 IDENTIFIED BY x CONTAINER=ALL;
```

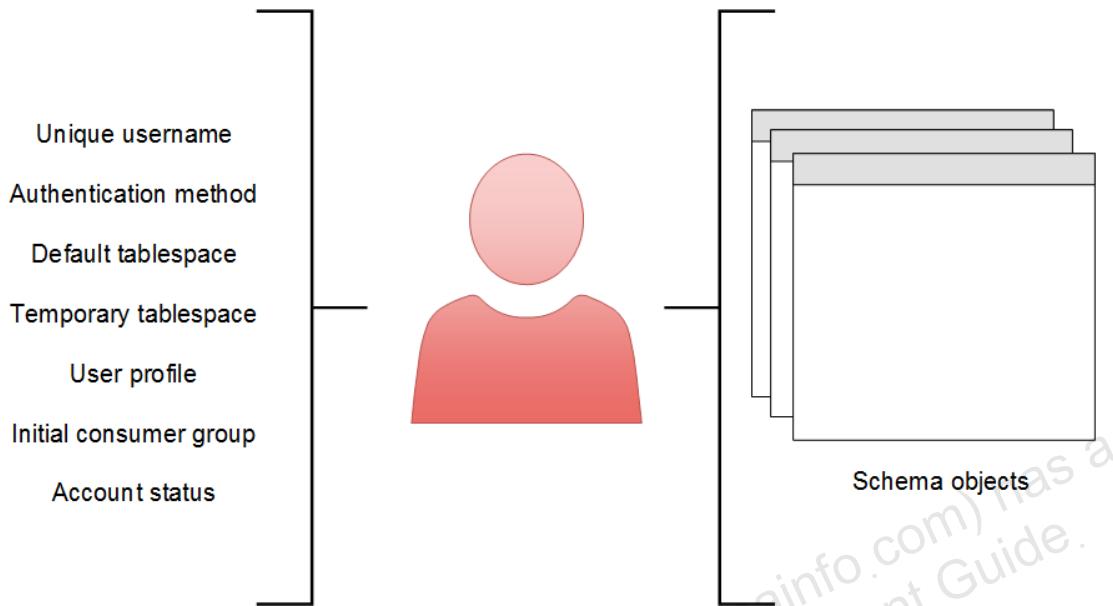
- **Local user:** The user is created in a single PDB only. Local users cannot be created in a root container nor in an application root container. A local user cannot create a common user. To create a local user, log in to the PDB where you want to create the local user, and issue the CREATE USER command, for example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA
SQL> CREATE USER l_user1 ... ;
```

In the illustration above, the common user `c##c_admin1` exists in the root container and all PDBs, while the local user `l_user1` only exists in `PDB1` and the local user `l_user2` only exists in `PDB2`.

You can use tools such as SQL*Plus and Enterprise Manager Database Express to create user accounts in the Oracle database.

User Accounts



ORACLE

User Account Components

To access the database, a user must specify a valid database user account and successfully authenticate as required by that user account. Each database user has a unique database account. Oracle recommends this to avoid potential security holes and provide meaningful data for certain audit activities. However, users may sometimes share a common database account. In these rare cases, the operating system and applications must provide adequate security for the database.

Each user account has the following, as illustrated in the diagram above:

- **Unique username:** Usernames cannot exceed 30 bytes, cannot contain special characters, and must start with a letter.
- **Authentication method:** The most common authentication method is a password.
- **Default tablespace:** This is a place where a user creates objects if the user does not specify some other tablespace.
 - Having a default tablespace does not imply that the user has the privilege of creating objects, nor does the user have a quota of space in that tablespace in which to create objects. Both of these privileges are granted separately.
 - If a user does not specify a tablespace when creating an object, the object will be created in the default tablespace assigned to the object owner. This enables you to control where the user's objects are created.
 - If an administrator does not define a default tablespace, the system-defined default permanent tablespace is used.

- Quota for a specific tablespace is not granted through a privilege. It's done by using the `ALTER USER` command, which changes the attributes for a user. However, if a DBA grants the `UNLIMITED TABLESPACE` system privilege to a user, then that user can use all the space in any tablespace.
- **Temporary tablespace:** This is a place where temporary objects, such as sorts and temporary tables, are created on behalf of the user by the instance. No quota is applied to temporary tablespaces. If an administrator does not define a temporary tablespace for a user, the system-defined temporary tablespace is used when the user creates objects.
- **User profile:** This is a set of resource and password restrictions assigned to the user.
- **Initial consumer group:** This is used by the Resource Manager.
- **Account status:** Users can access only “open” accounts. The account status may be “locked” and/or “expired.”

Note: A database user is not necessarily a person. It is a common practice to create a user that owns the database objects of a particular application, such as HR. The database user can be a device, an application, or just a way to group database objects for security purposes. The personal identifying information of a person is not needed for a database user.

Schemas

A *schema* is a collection of database objects that are owned by a database user. Schema objects are the logical structures that directly refer to the database’s data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, schema objects include everything that your application creates in the database.

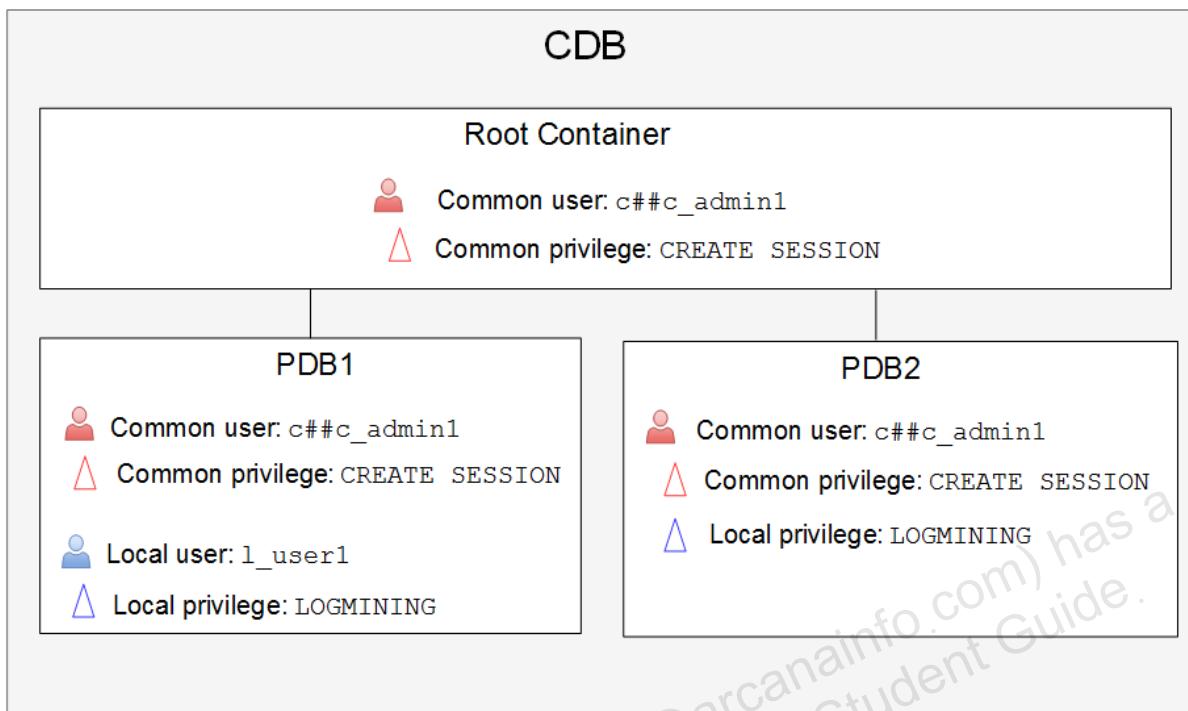
Samad Ul Haq (samad.ulhaq@arcanainfo.com),
non-transferable license to use this Student Guide.

Oracle-Supplied Administrator Accounts

Account	Description
SYS	Super user. Owns the data dictionary and the Automatic Workload Repository (AWR). Used for starting up and shutting down the database instance
SYSTEM	Owns additional administrative tables and views
SYSBACKUP	Facilitates Oracle Recovery Manager (RMAN) backup and recovery operations
SYSDG	Facilitates Oracle Data Guard operations
SYSKM	Facilitates Transparent Data Encryption wallet operations
SYSRAC	For Real Application Cluster (RAC) database administration tasks
SYSMAN	For Oracle Enterprise Manager database administration tasks
DBSNMP	Used by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database

ORACLE®

Granting Privileges



ORACLE

Common Versus Local Privileges

A **privilege** is a right to execute operations that create, modify, or delete structures in an instance or a database; or a right to execute operations that manipulate users' objects. Oracle Database has predefined system and object privileges. You do not create them. It is best to grant a privilege only to a user who requires that privilege to accomplish the necessary work. Excessive granting of unnecessary privileges can compromise security.

In a multitenant environment, you can grant a privilege to a user two ways:

- **Commonly:** You grant the user the privilege in all containers of a CDB. In an application container, a common privilege is granted to a grantee in the application root and application PDBs. To grant a privilege commonly, you must log in to the root container and issue the GRANT command with the CONTAINER=ALL clause, for example:

```
SQL> CONNECT / AS SYSDBA
SQL> GRANT create session TO c##c_admin1 CONTAINER=ALL;
```

- **Locally:** You grant the user the privilege in a single PDB only. To grant a privilege locally, log in to the PDB and issue the GRANT command, for example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA
SQL> GRANT logmining TO l_user1;
```

To switch to a different container, a common user must have the `SET CONTAINER` privilege in the current container. Alternatively, a common user can start a new database session whose initial current container is the container the user wants, relying on the `CREATE SESSION` privilege in that PDB. Be aware that commonly granted privileges that have been made to common users may interfere with the security configured for individual PDBs.

The diagram above illustrates the following things:

- The common user `c##c_admin1` is granted the privilege `CREATE SESSION` commonly, which applies the privilege to that user in all containers. In PDB2, `c##c_admin1` is also granted the `LOGMINING` privilege locally. `c##c_admin1` does not have the `LOGMINING` privilege in any other container.
- The local user `l_user1` exists in PDB1 only, and is granted the `LOGMINING` privilege. If the same user existed in another PDB (as a totally separate user), the `LOGMINING` privilege would not be applied.

Remember

Step 1: Create a common user when the same user has to perform the same actions in all PDBs in the CDB, otherwise create the user as a local user in a PDB.

Step 2: Ask yourself - do you want the common user who exists in each PDB to have the same privilege(s) in the PDBs?

- If yes, then you commonly grant the privilege(s) to the common user. Connect to the CDB as a user who is privileged enough to do it, and grant privilege1, privilege2, and so on to the common user by using the `CONTAINER=ALL` clause.
- If no, then you locally grant the privilege(s) to the common user. Connect to the PDB as a user who is privileged enough to do it, and grant privilege1, privilege2, and so on to the common user.

System Privileges

- Each system privilege allows a user to perform a particular database operation or class of database operations.
- Special system privileges for administrators:
 - SYSDBA
 - SYSOPER
 - SYSASM
 - SYSBACKUP
 - SYSDG
 - SYSKM
 - SYSRAC
- A system privilege with the ANY clause means the privilege applies to all schemas, not just your own.
- If you grant a system privilege with the ADMIN OPTION enabled, you enable the grantee to administer the system privilege and grant it to other users.

ORACLE®

Special System Privileges for Administrators

There are seven special system privileges that are usually granted only to administrators (listed in the table below). Anyone who is granted one of these privileges is referred to as a **system administrative privileged** user (privileged user, for short).

Administrator Privilege	Description
SYSDBA	You can perform all administrative tasks in the database, for example, you can create and drop a database, open and mount a database, start up and shut down an Oracle database, create an spfile, put a database in or remove a database from ARCHIVELOG mode, perform incomplete recovery operations, patch, and migrate. This privilege enables you to connect as the SYS user (this topic is discussed later in the lesson in more detail).
SYSOPER	You can perform similar administration tasks as the SYSDBA privilege, but without the ability to look at user data. For example, you can start up and shut down the database, create an spfile, and perform complete recovery operations (not incomplete recovery operations).
SYSASM	You can start up, shut down, and administer an Automatic Storage Management instance.
SYSBACKUP	You can perform Oracle Recovery Manager (RMAN) backup and recovery operations by using RMAN or SQL*Plus.
SYSDG	You can perform Data Guard operations by using the Data Guard Broker or the DGMGR command-line interface.
SYSKM	You can manage Transparent Data Encryption wallet operations.
SYSRAC	You can perform the day-to-day administration tasks on the Oracle Database for a Real Application Cluster (RAC) cluster.

Note:

- Only users who are granted the SYSDBA, SYSOPER, SYSASM , and SYSRAC privileges are allowed to start up and shut down the Oracle database.
- The SYSBACKUP, SYSDG, and SYSKM privileges enable you to connect to the database even if the database is not open.
- The SELECT ANY DICTIONARY system privilege does not permit access to sensitive data dictionary tables, which are owned by the SYS schema.

ANY Clause

Many system privileges contain an ANY clause, which means the privilege applies to all schemas, not just your own. For example, the SELECT ANY TABLE system privilege allows you to retrieve data from all tables and views, including those from schemas owned by other users. The SYS user and users with the DBA role are granted all of the ANY privileges; therefore, they can do anything to any data object. You can control the scope of all system privileges, including those with the ANY clause, by using the Oracle Database Vault Option.

ADMIN OPTION

If you grant a system privilege with the ADMIN OPTION enabled, you enable the grantee to administer the system privilege and grant it to other users. The SQL syntax for granting system privileges is:

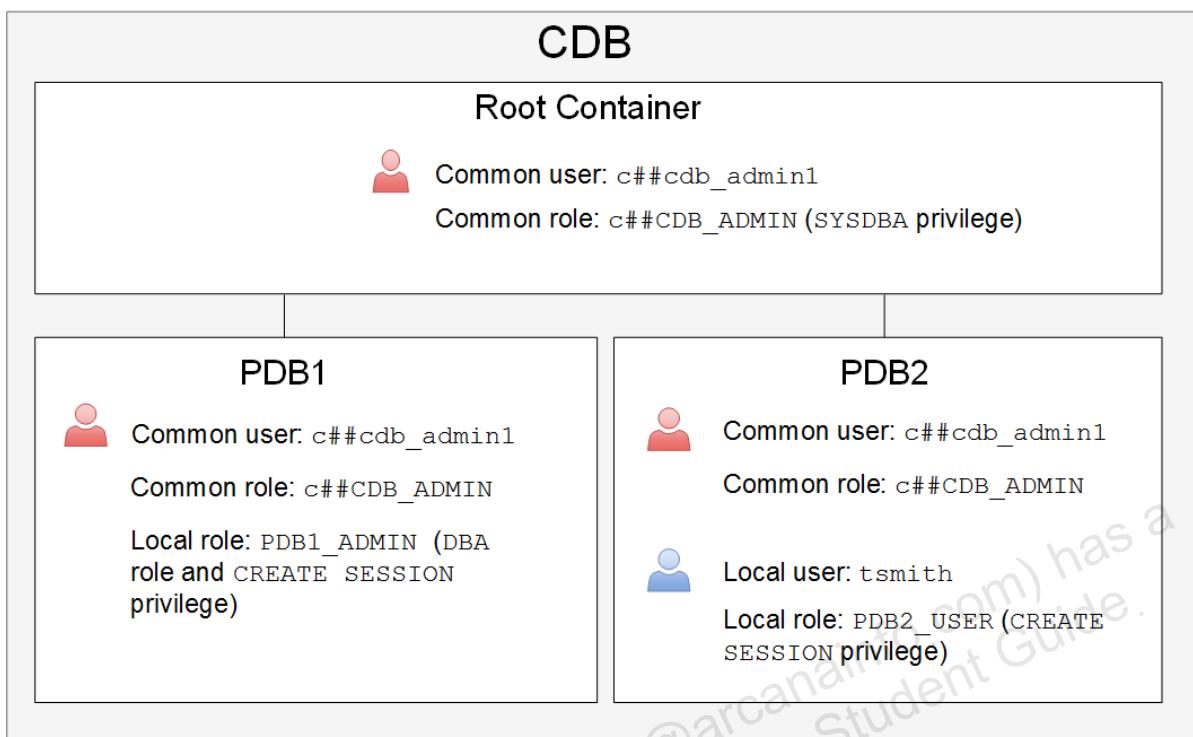
```
SQL> GRANT <system_privilege> TO <grantee_clause> [WITH ADMIN OPTION]
```

Object Privileges

- Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package.
- Without specific permission, users can access only their own objects.
- Object privileges can be granted by the owner of an object, by the administrator, or by someone who has been explicitly given permission to grant privileges on the object.
- The SQL syntax for granting object privileges is:

```
SQL> GRANT <object_privilege> ON <object> TO <grantee_clause> [WITH GRANT  
OPTION]
```

Creating and Granting Roles



ORACLE

Creating Roles

A **role** is a group of privileges and/or other roles, and is granted to a user or a role. Roles are usually created by administrators; however, there are several Oracle-supplied roles. You can use several tools, including SQL*Plus and Enterprise Manager Express, to create and grant roles.

Roles provide the following benefits with respect to managing privileges:

- Easier privilege management: Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role and then grant that role to each user.
- Dynamic privilege management: If the privileges associated with a role are modified, all users who are granted the role acquire the modified privileges automatically and immediately.
- Selective availability of privileges: Roles can be enabled and disabled to turn privileges on and off temporarily. This allows the privileges of the user to be controlled in a given situation.

In a multitenant environment, you can create two types of roles:

- **Common role:** The role is replicated in all current and future containers. All Oracle-supplied predefined roles are common roles. To create a common role, connect to the root container, and issue the `CREATE ROLE` command with the `CONTAINER=ALL` clause appended to end. Local users cannot create common roles. For example:

```
SQL> CONNECT / AS SYSDBA
SQL> CREATE ROLE c##c_role1 CONTAINER=ALL;
```

- **Local role:** The role is created in a single PDB, and can be used within that PDB only. It does not have any commonly granted privileges. To create a local role, you must connect to the container, and issue the CREATE ROLE command, for example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA
SQL> CREATE ROLE l_role1;
```

The diagram above illustrates the following things:

- A common role named c##CDB_ADMIN is created in all containers. This role consists of the SYSDBA privilege and is created for users who need to perform maintenance operations on the entire CDB. The common user c##c_admin1, who exists in every container, is granted the c##CDB_ADMIN role commonly; meaning, c##c_admin1 is granted that role in every container.
- A local role named PDB1_ADMIN is created and available in PDB1 only. This role consists of the DBA role and the CREATE SESSION privilege, and is created for users who manage PDB1. The common user named c##c_admin1 is granted this role only in PDB1.
- A local role named PDB2_USER is created and available in PDB2 only. The local user named tsmith, who exists only in PDB2, is granted the PDB2_USER role.

Role Characteristics

Role characteristics include:

- Privileges are granted to and revoked from roles as though the role were a user.
- Roles are granted to and revoked from users or other roles as though they were system privileges.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Roles are not owned by anyone, and they are not in any schema.
- The SYSDBA privilege is not granted in any role, so you grant it (either commonly or locally) to a user or role.

Granting Roles

- To grant a role to a user or another role by using SQL*Plus, use the `GRANT` command.
- Two ways to grant a role:
 - Commonly: Grant the role to the user (or role) in all containers.

```
SQL> CONNECT / AS SYSDBA  
SQL> GRANT <common role> TO <common user or role> CONTAINER=ALL;
```

- Locally: Grant the role to a user (or role) in one PDB only.

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> GRANT <common or local role> TO <common or local user>;
```

ORACLE

Granting Roles Commonly Versus Locally

There are two ways to grant a role:

- **Commonly:** You grant the role to the user (or role) in all containers. Do this if the user needs to perform the same operation in all containers. To grant a role commonly, log in to the root container, and issue the `GRANT` command to a common user (or common role) with the `CONTAINER=ALL` clause. The role that you grant must be a common role before you can grant it commonly. For example:

```
SQL> CONNECT / AS SYSDBA  
SQL> GRANT <common role> TO <common user or role> CONTAINER=ALL;
```

- **Locally:** You grant the role to a user (or role) in one PDB only. To grant a role locally, log in to the PDB where the user exists and issue the `GRANT` command *without* the `CONTAINER=ALL` clause. The role that you grant can be a common or local role. For example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> GRANT <common or local role> TO <common or local user>;
```

Oracle-Supplied Roles

This table lists some commonly used predefined roles in Oracle Database:

Role	Privileges Included
DBA	Includes most system privileges and several other roles. Do not grant this role to non-administrators. Users with this role can connect to the CDB or PDB only when it is open.
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
SCHEDULER_ADMIN	CREATE ANY JOB, CREATE EXTERNAL JOB, CREATE JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER
SELECT_CATALOG_ROLE	SELECT privileges on data dictionary objects



Making Roles More Secure

- Roles are usually enabled by default, which means that if a role is granted to a user, then that user can exercise the privileges given to the role immediately.
- Default roles are assigned to the user at connect time.
- Use the following security measures to make roles more secure:
 - Make a role non-default.
 - Use role authentication.
 - Create application roles.

ORACLE®

- **Create application roles.** Create secure application roles that a user must enable by executing a PL/SQL procedure successfully. The PL/SQL procedure can check things, such as the user's network address, the program that the user is running, the time of day, and other elements needed, to properly secure a group of permissions.

```
SQL> CREATE ROLE secure_application_role IDENTIFIED USING  
<security_procedure_name>;
```

Revoking Privileges and Roles

Unauthorized reproduction or distribution prohibited
Copyright © 2017, Oracle and/or its affiliates.

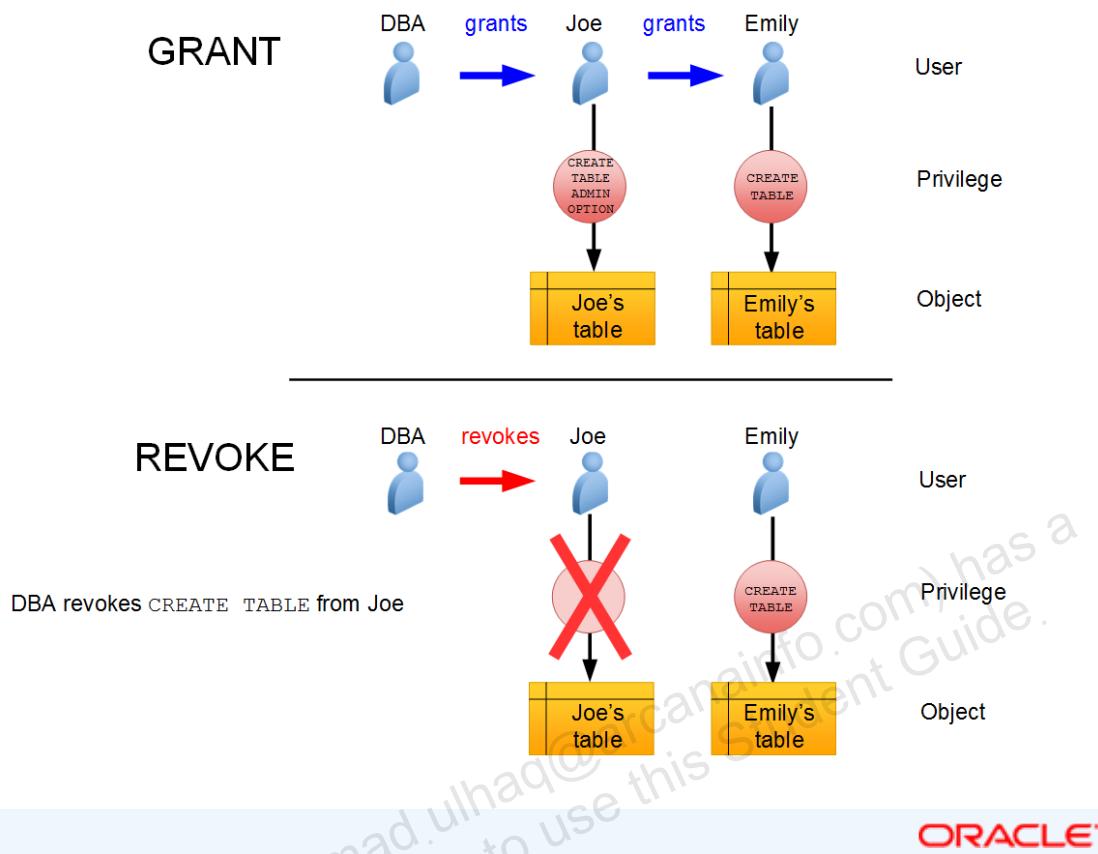
You can use the `REVOKE` statement to:

- Revoke system privileges from users and roles
- Revoke roles from users, roles, and program units
- Revoke object privileges for a particular object from users and roles

ORACLE

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Revoking System Privileges



System privileges that have been granted directly with a GRANT command can be revoked by using the REVOKE command in SQL*Plus. Users with the ADMIN OPTION for a system privilege can revoke the privilege from any other database user. The revoker does not have to be the same user who originally granted the privilege.

There are no cascading effects when a system privilege is revoked, regardless of whether it is given the ADMIN OPTION.

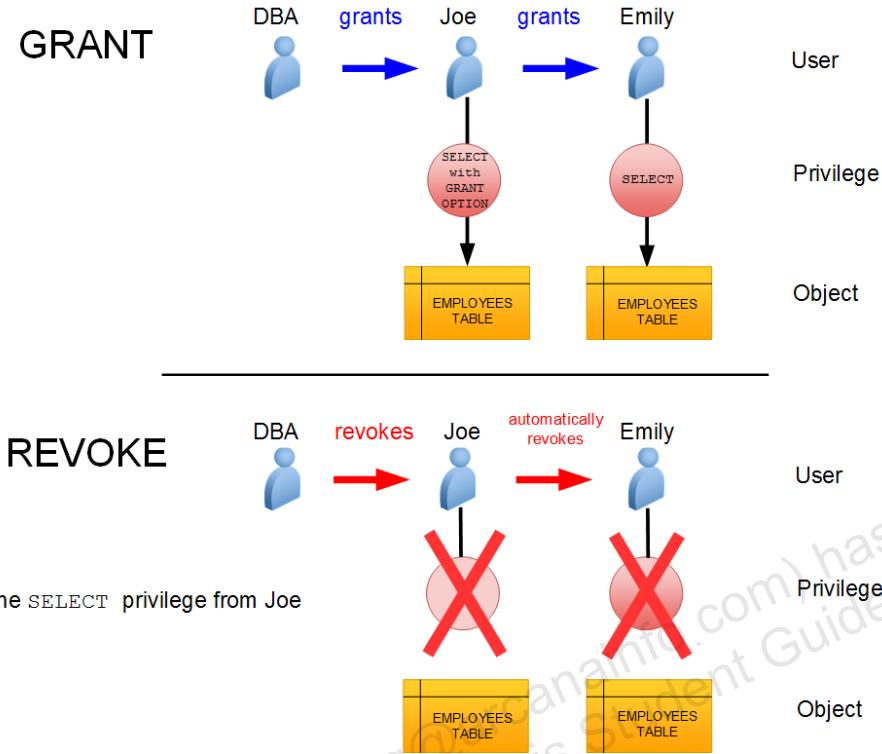
The SQL syntax for revoking system privileges is:

```
SQL> REVOKE <system_privilege> FROM <grantee_clause>
```

The diagram above illustrates the following events:

1. The DBA grants the CREATE TABLE system privilege to the user Joe with ADMIN OPTION.
2. Joe creates a table.
3. Joe grants the CREATE TABLE system privilege to the user Emily.
4. Emily creates a table.
5. The DBA revokes the CREATE TABLE system privilege from Joe.
6. The result is that Joe's table still exists and he can still access it, but he can't create new tables. Emily's table still exists, and she still has the CREATE TABLE system privilege.

Revoking Object Privileges



Cascading effects can be observed when revoking a system privilege that is related to a data manipulation language (DML) operation. For example, if the `SELECT ANY TABLE` privilege is granted to a user, and if that user has created procedures that use the table, all procedures that are contained in the user's schema must be recompiled before they can be used again.

Revoking object privileges also cascades when given with `GRANT OPTION`. As a user, you can revoke only those privileges that you have granted. For example, Bob cannot revoke the object privilege that Joe granted to Emily. Only the grantee or a user with the privilege called `GRANT ANY OBJECT PRIVILEGE` can revoke object privileges.

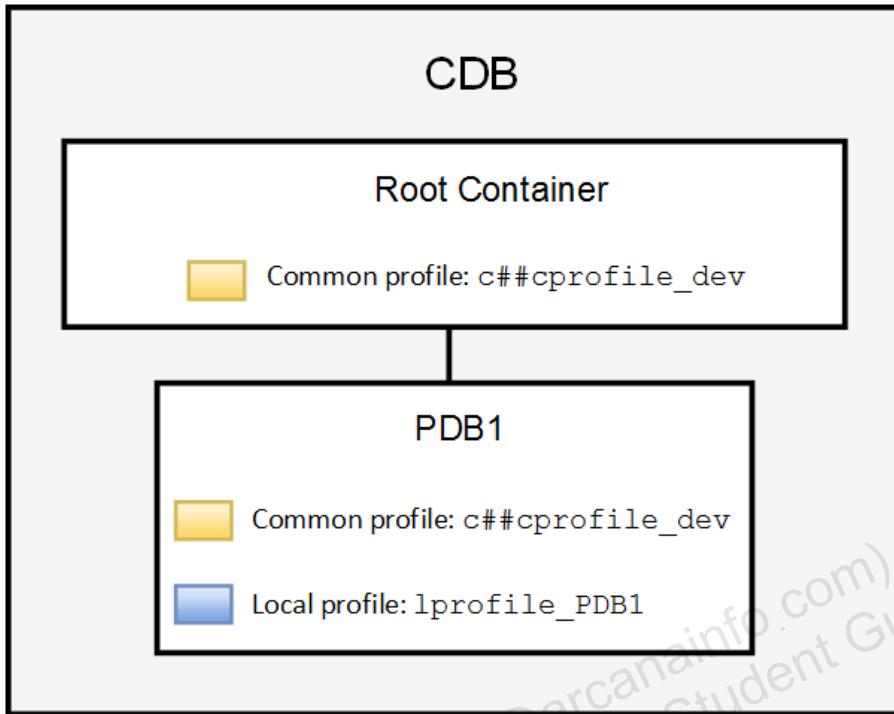
The diagram above illustrates object privileges being revoked. Suppose that the following events occur:

1. A DBA grants Joe the `SELECT` object privilege on the `EMPLOYEES` table with the `GRANT OPTION`.
2. Joe grants the `SELECT` privilege on the `EMPLOYEES` table to Emily.
3. The DBA revokes the `SELECT` privilege from Joe.
4. The result is that the revoke takes away Joe's ability to access the `EMPLOYEES` table, and the revoke is cascaded to Emily as well.

Revoking Roles

- To revoke a role from a user or another role:
 - You must have been directly granted the role with the ADMIN OPTION, or
 - You must have created the role.
- You can revoke any role if you have the GRANT ANY ROLE system privilege.

Creating and Assigning Profiles



ORACLE

About Profiles

A **profile** is a named set of resource limits and password parameters that restrict database usage and database instance resources for a user. If you assign a profile to a user, then that user cannot exceed those limits. Every user, including administrators, is assigned a profile and may belong to only one profile at any given time. By default, when users are created, they are assigned the **DEFAULT** profile unless another profile is specified.

Creating Profiles

Oracle recommends that you use profiles to create and manage password security (you can use the **CREATE PROFILE** command), and use Database Resource Manager to manage machine resources because it offers a more flexible means of managing and tracking resource use.

In a multitenant environment, you can create two types of profiles:

- **Common profile:** The profile is replicated in all current and future containers. To create a common profile by using SQL*Plus, log in to the root container and issue the **CREATE PROFILE** command with the **CONTAINER=ALL** clause, for example:

```
SQL> CONNECT / AS SYSDBA
SQL> CREATE PROFILE c##cprofile_dev limit ... CONTAINER=ALL;
```

- **Local profile:** The profile is created in a single PDB, and can be used within that PDB only. To create a local

profile by using SQL*Plus, log in to the PDB and issue the CREATE PROFILE command without the CONTAINER=ALL clause, for example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA
SQL> CREATE PROFILE lprofile_PDB1 limit ... ;
```

In the diagram above, the common profile named c##cprofile_dev is created commonly at the CDB level. The CREATE operation is replicated in all containers including the root container where it was initially created. Consequently, the same profile c##cprofile_dev is created in PDB1. The profile named lprofile_PDB1 is created locally in PDB1 and exists only in PDB1.

Dropping or Changing Profiles

In Enterprise Manager Database Express, you cannot drop a profile that is used by users. However, if you drop a profile with the CASCADE option (for example, in SQL*Plus), all users who have that profile are automatically assigned the DEFAULT profile.

If users have already logged in when you change their profile, the change does not take effect until their next login.

Assigning Profiles

There are two ways to assign a profile:

- **Commonly:** The profile assignment is replicated in all current and future containers.

```
SQL> CONNECT / AS SYSDBA  
SQL> ALTER USER <common user> PROFILE <common profile> CONTAINER=ALL;
```

- **Locally:** The profile assignment occurs in one PDB (stand-alone or application container) only.

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> ALTER USER <common or local user> PROFILE <common or local profile>;
```

ORACLE

Password Parameters

- In a profile, specific parameters control account locking, password aging and expiration, and password history.
 - Profile password settings are always enforced.
- **Account locking** enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts or when accounts sit inactive for a pre-defined number of days (meaning, users have not attempted to log in to their accounts).
- **Password aging and expiration** enables user passwords to have a lifetime, after which the passwords expire and must be changed.
- **Password history** checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes.
- **Password complexity verification** makes a complexity check on the password to verify that it meets certain rules.

ORACLE®

Account Locking

Account locking enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts or when accounts sit inactive for a pre-defined number of days (meaning, users have not attempted to log in to their accounts). Configure the following profile parameters:

- FAILED_LOGIN_ATTEMPTS specifies the number of failed login attempts before the lockout of the account.
- PASSWORD_LOCK_TIME specifies the number of days for which the account is locked after the specified number of failed login attempts.
- INACTIVE_ACCOUNT_TIME specifies the number of days an account can be inactive before it is locked.

Password Aging and Expiration

Password aging and expiration enables user passwords to have a lifetime, after which the passwords expire and must be changed. Configure the following profile parameters:

- PASSWORD_LIFE_TIME determines the lifetime of the password in days, after which the password expires.
- PASSWORD_GRACE_TIME specifies a grace period in days for changing the password after the first successful login after the password has expired.

Eating passwords and locking the SYS, SYSMAN, and DBSNMP accounts prevent Enterprise Manager from functioning properly. The applications must catch the “password expired” warning message and handle the password change; otherwise, the grace period expires and the user is locked out without knowing the reason.

Password History

Password history checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes. Configure one of the following parameters:

- `PASSWORD_REUSE_TIME` specifies that a user cannot reuse a password for a given number of days.
- `PASSWORD_REUSE_MAX` specifies the number of password changes that are required before the current password can be reused.
- `PASSWORD_VERIFY_FUNCTION` checks for password complexity for the `SYS` user.

Recall that the values of the profile parameters are either set or inherited from the `DEFAULT` profile.

If both password history parameters have a value of `UNLIMITED`, Oracle Database ignores both. The user can reuse any password at any time, which is not a good security practice. If both parameters are set, password reuse is allowed, but only after meeting both conditions. The user must have changed the password the specified number of times, and the specified number of days must have passed since the old password was last used. For example, the profile of user `ALFRED` has `PASSWORD_REUSE_MAX` set to 10 and `PASSWORD_REUSE_TIME` set to 30. User `ALFRED` cannot reuse a password until he has reset the password 10 times and until 30 days have passed since the password was last used. If one parameter is set to a number and the other parameter is specified as `UNLIMITED`, then the user can never reuse a password.

Password Complexity Verification

Password complexity verification makes a complexity check on the password to verify that it meets certain rules. The check must ensure that the password is complex enough to provide protection against intruders who may try to break into the system by guessing the password.

The `PASSWORD_VERIFY_FUNCTION` parameter names a PL/SQL function that performs a password complexity check before a password is assigned. Password verification functions must be owned by the `SYS` user and must return a Boolean value (`TRUE` or `FALSE`). A model password verification function is provided in the `ut1pwdmg.sql` script found in the following directories:

- UNIX and Linux platforms: `$ORACLE_HOME/rdbms/admin`
- Windows platforms: `%ORACLE_HOME%\rdbms\admin`

You can create complexity functions and run the `ut1pwdmg.sql` script to create the `VERIFY_FUNCTION_11G`, `ORA12C_VERIFY_FUNCTION`, and `ORA12C_STRONG_VERIFY_FUNCTION` pre-built complexity functions.

Resource Parameters

- In a profile, you can control:
 - CPU resources - may be limited on a per-session or per-call basis
 - Network and memory resources - you can specify the following:
 - Connect time
 - Idle time
 - Concurrent sessions
 - Private SGA
 - Disk I/O resources - limit the amount of data a user can read at the per-session level or per-call level
- Profiles cannot impose resource limitations on users unless the `RESOURCE_LIMIT` initialization parameter is set to TRUE.
 - With `RESOURCE_LIMIT` at its default value of FALSE, profile resource limitations are ignored.
- Profiles also allow *composite limits*, which are based on weighted combinations of CPU/session, reads/session, connect time, and private SGA.

ORACLE®

CPU Resources

CPU resources may be limited on a per-session or per-call basis. A CPU/Session limitation of 1,000 means that if any individual session that uses this profile consumes more than 10 seconds of CPU time (CPU time limitations are in hundredths of a second), that session receives the following error message and is logged off:

ORA-02392: exceeded session limit on CPU usage, you are being logged off.

A per-call limitation does the same thing, but instead of limiting the user's overall session, it prevents any single command from consuming too much CPU. If CPU/Call is limited and the user exceeds the limitation, the command aborts. The user receives an error message, such as:

ORA-02393: exceeded call limit on CPU usage

Network and Memory Resources

Each database session consumes system memory resources and (if the session is from a user who is not local to the server) network resources. You can specify the following:

- Connect Time: Indicates for how many minutes a user can be connected before being automatically logged off
- Idle Time: Indicates for how many minutes a user's session can remain idle before being automatically logged off. Idle time is calculated for the server process only. It does not take into account application activity. The

`IDLE_TIME` limit is not affected by long-running queries and other operations.

- Concurrent Sessions: Indicates how many concurrent sessions can be created by using a database user account
- Private SGA: Limits the amount of space consumed in the System Global Area (SGA) for sorting, merging bitmaps, and so on. This restriction takes effect only if the session uses a shared server configuration.

Disk I/O Resources

Disk I/O resources limit the amount of data a user can read at the per-session level or per-call level. `Reads/Session` and `Reads/Call` place a limitation on the total number of reads from both memory and the disk. This can be done to ensure that no I/O-intensive statements overuse memory and disks.

Oracle-Supplied Password Verification Functions

- Complexity verification checks that each password is complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords.
- You can create your own password verification functions.
- Oracle Database provides the following functions that you can create by executing the `utlpwdmg.sql` script:
 - `ORA12C_VERIFY_FUNCTION`
 - `ORA12C_STRONG_VERIFY_FUNCTION`
 - `VERIFY_FUNCTION_11G`
- The functions above must be owned by the `SYS` user.
 - Password complexity checking is not enforced for the `SYS` user.

ORACLE

In addition to creating the functions, the `utlpwdmg.sql` script also changes the `DEFAULT` profile with the following `ALTER PROFILE` command:

```
ALTER PROFILE default LIMIT  
PASSWORD_LIFE_TIME 180  
PASSWORD_GRACE_TIME 7  
PASSWORD_REUSE_TIME UNLIMITED  
PASSWORD_REUSE_MAX UNLIMITED  
FAILED_LOGIN_ATTEMPTS 10  
PASSWORD_LOCK_TIME 1  
PASSWORD_VERIFY_FUNCTION ora12c_verify_function;
```

Authenticating Users

- Every user, including administrators, must be authenticated when connecting to a database instance.
 - Authentication verifies that the user is a valid database user, and establishes a trust relationship for further interactions.
 - Authentication also enables accountability by making it possible to link access and actions to specific identities.
- The following authentication methods are possible:
 - Password (usually for database users)
 - Operating system (OS) authentication
 - Password file (for system administrative privileged users only)
 - Strong authentication with Kerberos, SSL, or directory authentication
- A system administrative privileged user must use OS authentication, password file authentication, or strong authentication.
 - These methods can authenticate when the database is available or unavailable (not started).

ORACLE

Your choice of authentication is influenced by whether you intend to administer your database locally on the same system where the database resides, or whether you intend to administer many different databases from a single remote client.

View:

Examine the following diagram to help you to choose an authentication method.

- [Authentication Methods for Database Administrators](#) (scroll down to Figure 1-2 Database Administrator Authentication Methods)

To connect to Oracle Database as a system administrative privileged user over a nonsecure connection, you must be authenticated by a password file.

To connect to Oracle Database as a system administrative privileged user over a local connection or a secure remote connection, choose one of the following:

- If the database has a password file and you have been granted a system privilege, then you can connect and be authenticated by a password file.

- If the database does not have a password file, or if you have not been granted a system privilege and are therefore not in the password file, then you can use operating system authentication. On most operating systems, authentication for database administrators involves placing the operating system username of the database administrator in a special group.

Important: If a system administrative privileged user is configured in both the password file and OS group, OS authentication takes precedence over password file authentication.

Password Authentication

- Password authentication is also referred to as "authentication" by the Oracle Database server.
- Create each user with an associated password that must be supplied when the user attempts to establish a connection.
- When setting up a password, you can expire the password immediately, which forces the user to change the password after first logging in.
 - If you decide on expiring user passwords, make sure that users have the ability to change the password.
 - Some applications do not have this functionality.
 - All passwords created in Oracle Database are case-sensitive by default.
 - Passwords may contain multibyte characters and are limited to 30 bytes.
- Passwords are always automatically and transparently encrypted by using the Advanced Encryption Standard (AES) algorithm during network (client/server and server/server) connections before sending them across the network.

Password File Authentication

- You can use password file authentication for an Oracle database instance and for an Oracle Automatic Storage Management (Oracle ASM) instance.
 - If authentication succeeds, the connection is logged with the SYS user.
- A password file stores database user names and case-sensitive passwords for administrator users (common and local administrators).
 - DBCA creates a password file during installation.
- To prepare for password file authentication, you must:
 - Create the password file.
 - Set the REMOTE_LOGIN_PASSWORDFILE initialization parameter.
 - Grant system administrative privileges (for example, GRANT SYSDBA to mydba)
- Use the CONNECT command in SQL*Plus to connect, for example:

```
SQL> CONNECT mydba AS SYSDBA
```



For more information, see the following sources in *Oracle Database Administrator's Guide*:

- [Preparing to Use Password File Authentication](#)
- [Connecting Using Password File Authentication](#)

On Unix and Linux, the password file is called `orapw ORACLE_SID` and is stored in `$ORACLE_HOME/dbs`. On Windows, the file is called `PWD ORACLE_SID.ora` and is stored in `$ORACLE_HOME\database`.

If your concern is that the password file might be vulnerable or that the maintenance of many password files is a burden, strong authentication can be implemented. You can query the `V$PFILE_USERS` view information in the password file.

OS Authentication

- Oracle Universal Installer creates operating system groups, assigns them specific names, and maps each group to a specific system privilege.
 - Example: Members of the dba group are granted SYSDBA
- As a group member, you can be authenticated, enabled as an administrative user, and connected to a local database:

```
SQL> CONNECT / AS SYSDBA
SQL> CONNECT / AS SYSOPER
SQL> CONNECT / AS SYSBACKUP
SQL> CONNECT / AS SYSDG
SQL> CONNECT / AS SYSKM
SQL> CONNECT / AS SYSRAC
```

- If you are not a member of one of these OS groups, you will not be able to connect as an administrative user via OS authentication.

ORACLE

OS Group	Unix or Linux User Group	Special System Privilege Granted to Members of the User Group
Oracle Software Group (top level group)	oinstall	Allowed to create and delete database files on the OS All database administrators belong to this group.
Database Administrator Group (OSDBA)	dba	SYSDBA Connects you as the SYS user
Database Operator Group (OSOPER) - optional	oper	SYSOPER Connects you as the PUBLIC user
Database Backup and Recovery Group (OSBACKUPDBA)	backupdba	SYSBACKUP
Data Guard Administrative Group (OSDGDBA)	dgdba	SYSDG
Encryption Key Management Administrative Group (OSKMDBA)	kmdba	SYSKM
Real Application Cluster Administrative Group (OSRACDBA)	rac	SYSRAC

If you are not a member of one of the above OS groups, you will not be able to connect as an administrative user via OS authentication. That is, CONNECT / AS SYSDBA will fail. However, you can still connect using other authentication methods, for example, network, password, or directory-based authentication.

Example

The special system privileges are not exercised unless you include them in your CONNECT clause. For example, suppose the HR user is granted the SYSDBA privilege and connects with that privilege. Notice that the current user becomes SYS:

```
SQL> CONNECT hr/hr@PDB1 AS SYSDBA
Connected.
SQL> SHOW USER
USER is "SYS"
```

However, if the HR user logs in to PDB1 without including the AS SYSDBA clause, the current user is HR and the user does not have the SYSDBA privilege.

```
SQL> CONNECT hr/hr@PDB1
Connected.
SQL> SHOW USER
USER is "HR"
```

OS Authentication for Users

If your operating system permits, you can have it authenticate users. They will not need to provide a username or password when connecting to the database instance.

If you use operating system authentication, set the `OS_AUTHENT_PREFIX` initialization parameter and use this prefix in Oracle usernames. The `OS_AUTHENT_PREFIX` parameter defines a prefix that the Oracle database adds to the beginning of each user's operating system account name. The default value of this parameter is `OPS$` for backward compatibility with the previous versions of the Oracle software. The Oracle database compares the prefixed username with the Oracle usernames in the database when a user attempts to connect. For example, suppose that `OS_AUTHENT_PREFIX` is set as follows:

```
OS_AUTHENT_PREFIX=OPS$
```

If a user with an operating system account named `tsmith` needs to connect to an Oracle database and be authenticated by the operating system, the Oracle database checks whether there is a corresponding database user `OPS$tsmith` and, if so, allows the user to connect. All references to a user who is authenticated by the operating system must include the prefix, as seen in `OPS$tsmith`. The text of the `OS_AUTHENT_PREFIX` initialization parameter is case-sensitive on some operating systems.

Assigning Quotas to Users

- A *quota* is a space allowance in a given tablespace.
- By default, a user has no quota on any of the tablespaces.
- Database accounts that need quota are those that own database objects.
 - Example: accounts for applications
- Only those activities that use space in a tablespace count against quota.
 - Oracle server checks quota when you create or extend a segment.
 - Activities that don't use space don't impact quota (example: create view).
 - You can be granted permission to use objects without needing any quota.
- Quota is not needed for assigned temporary tablespaces or undo tablespaces.
- A user's quota is replenished when he drops objects (with the PURGE clause) or purges his objects in the recycle bin.
- You have three options for providing quota for a user on a tablespace:
 - Unlimited
 - Value
 - UNLIMITED TABLESPACE system privilege

ORACLE

You have three options for providing a quota for a user on a tablespace.

- **Unlimited:** Allows the user to use as much space as is available in the tablespace
- **Value:** Number of kilobytes or megabytes that the user can use. This does not guarantee that the space is set aside for the user. This value can be larger or smaller than the current space that is available in the tablespace.
- **UNLIMITED TABLESPACE system privilege:** Overrides all individual tablespace quotas and gives the user unlimited quota on all tablespaces, including SYSTEM and SYSAUX. This privilege must be granted with caution.

You must not provide a quota to users on the SYSTEM or SYSAUX tablespaces. Typically, only the SYS and SYSTEM users are able to create objects in the SYSTEM or SYSAUX tablespaces.

Applying the Principal of Least Privilege

- The *principle of least privilege* means that a user must be given only those privileges that are required to efficiently complete a task.
 - This reduces the chances of users modifying or viewing data (either accidentally or maliciously) that they do not have the privilege to modify or view.
- Ways to apply the principal of least privilege:
 - Protect the data dictionary
 - Revoke unnecessary privileges from PUBLIC
 - Use access control lists (ACLs) to control network access
 - Restrict access to OS directories
 - Limit users with administrative privileges
 - Restrict remote database authentication
 - Unified auditing

ORACLE®

Ways to Apply the Principal of Least Privilege

The following are ways that you can apply the principal of least privilege.

Protect the data dictionary:

The O7_DICTIONARY_ACCESSIBILITY parameter is set by default to FALSE. You must not allow this to be changed without a very good reason because it prevents users with the ANY TABLE system privileges from accessing the data dictionary base tables. It also ensures that the SYS user can log in only as SYSDBA.

Revoke unnecessary privileges from PUBLIC:

Several packages are extremely useful to applications that need them, but require proper configuration to be used securely. PUBLIC is granted execute privilege on the following packages: UTL_SMTP, UTL_TCP, UTL_HTTP, and UTL_FILE.

Use access control lists (ACLs) to control network access:

In Oracle Database, network access is controlled by an ACL that may be configured to allow certain users access to specific network services. Network access is denied by default. An ACL must be created to allow network access. File access through UTL_FILE is controlled at two levels:

- At the OS level with permissions on files and directories

- In the database by DIRECTORY objects that allow access to specific file system directories. The DIRECTORY object may be granted to a user for read or for read and write. Execute privileges on other PL/SQL packages should be carefully controlled.

The more powerful packages that may potentially be misused include:

- UTL_SMTP: Permits arbitrary email messages to be sent by using the database as a Simple Mail Transfer Protocol (SMTP) mail server. Use the ACL to control which machines may be accessed by which users.
- UTL_TCP: Permits outgoing network connections to be established by the database server to any receiving or waiting network service. Thus, arbitrary data can be sent between the database server and any waiting network service. Use the ACL to control access.
- UTL_HTTP: Allows the database server to request and retrieve data via HTTP. Granting this package to a user may permit data to be sent via HTML forms to a malicious website. Limit access by using the ACL.
- UTL_FILE: If configured improperly, allows text-level access to any file on the host operating system. When properly configured, this package limits user access to specific directory locations.

Restrict access to OS directories:

The DIRECTORY object inside the database enables DBAs to map directories to OS paths and to grant privileges on those directories to individual users.

Limit users with administrative privileges:

Do not provide database users more privileges than necessary. Non-administrators must not be granted the DBA role. To implement least privilege, restrict the following types of privileges:

- Grants of system and object privileges
- SYS-privileged connections to the database, such as SYSDBA and SYSOPER
- Other DBA-type privileges, such as DROP ANY TABLE

Restrict remote database authentication:

The REMOTE_OS_AUTHENT parameter is set to FALSE by default. It must not be changed unless all clients can be trusted to authenticate users appropriately. In the remote authentication process:

- The database user is authenticated externally
- The remote system authenticates the user
- The user logs in to the database without further authentication

Note: Always test your applications thoroughly if you have revoked privileges.

After having applied the principle of least privilege, you can track changes that users make in the database.

Unified auditing:

- Auditing is the monitoring and recording of configured database actions, from both database users.
- Unified Auditing centralizes all audit records in one place.
- You can configure auditing for both successful and failed activities, and include or exclude specific users from the audit.
- In a multitenant environment, you can audit individual actions of PDBs or individual actions in the entire CDB.
- Auditing is enabled by default for the SYS user (instance parameter audit_sys_operations = TRUE).
- All audit records are written to the unified audit trail in a uniform format and are made available through the UNIFIED_AUDIT_TRAIL view.

Summary for Lesson 5

In this lesson, you should have learned how to:

- Create database users
- Grant privileges to database users
- Create and grant roles to users or other roles
- Revoke privileges and roles from users and other roles
- Create and assign profiles to users
- Explain the various authentication options for users
- Assign quota to users
- Apply the principal of least privilege



ORACLE

Practice 5 Overview

- 5-1: Creating Common and Local Users
- 5-2: Creating a Local User for an Application in SQL*Plus
- 5-3: Granting a Local Role (DBA) to PDBADMIN in SQL*Plus
- 5-4: Creating a Local Profile in EM Express
- 5-5: Creating Local Roles in EM Express
- 5-6: Creating Local Users in EM Express and Script
- 5-7: Configuring a Default Role for a User
- 5-8: Auditing User Activity
- 5-9: Exploring OS and Password File Authentication

Practice 5-1 Creating Common and Local Users

Overview

In this practice, you log on to the database in SQL*Plus as the `SYS` user and create two types of administrators:

- **CDB administrator named `c##CDB_ADMIN1`:** Create this user as a common user so that it exists in every container in the CDB. Grant this user the most powerful administrator privilege, the `SYSDBA` privilege, in all containers. This privilege enables `c##CDB_ADMIN1` to access containers whether they are open or not. Because most database operations don't require the `SYSDBA` privilege, also grant this user the `DBA` role and `CREATE SESSION` privilege in all containers so that the user can operate as a regular user too.
- **PDB1 administrator named `PDB1_ADMIN1`:** Create this user as a local user in `PDB1`, and grant this user the `DBA` role and `CREATE SESSION` privilege. This grant will provide the necessary system and object privileges. All tasks required by this user must be performed on an open PDB.

Tip:

It's good practice to create a user separate from `SYS` and `SYSTEM` to perform database administration tasks. Each DBA in your organization should have his or her own privileged account to aid in auditing. Keep in mind that when you connect with the `SYSDBA` privilege, the database shows you logged in as the `SYS` user, regardless of your actual username. Audit trails, however, will show your real username.

Organizations that need to implement the tightest security possible, separate the database duties and create many accounts for each Database Administrator (DBA) distinctly named, and use the security principle of least privileges. Only the minimum amount of privileges needed to perform a job are given. If an administrator doesn't need access to data but still performs maintenance operations, you can grant that user the `SYSOPER` privilege instead. Also consider other administrative privileges, such as `SYSDG`, `SYSKM`, `SYSBACKUP` and `SYSRAC`.

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Create `c##CDB_ADMIN1`

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

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

```
$ sqlplus / as sysdba
```

```
SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
```

Connected to:

```
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

3. Create a common user named c##CDB_ADMIN1 by using the CREATE USER command. See Appendix - Product-Specific Credentials for the password. Assign c##CDB_ADMIN1 to the users tablespace and the temp temporary tablespace. Also unlock the account so that c##CDB_ADMIN1 can log in right away. Important! To create a common user, you must start the user name with c## or C## and you must include the CONTAINER=ALL clause so that the user's identity and password are created in all the containers.

```
SQL> CREATE USER c##CDB_ADMIN1 IDENTIFIED BY <password> CONTAINER=ALL DEFAULT
TABLESPACE users TEMPORARY TABLESPACE temp ACCOUNT UNLOCK;
```

User created.

```
SQL>
```

4. 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>
```

5. Question: Would the following statement complete the same operation?

```
GRANT create session, dba TO c##CDB_ADMIN1;
```

Answer: No, because without the CONTAINER=ALL clause, the CREATE SESSION privilege and DBA role are granted *locally* (in the root container only) to c##CDB_ADMIN1, and not to each c##CDB_ADMIN1 user in each PDB.

6. List the common users by querying the DBA_USERS view. Scroll down and verify that c##CDB_ADMIN is included at the end of the list.

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

USERNAME
-----
...
SI_INFORMN_SCHEMA
WMSYS
REMOTE_SCHEDULER_AGENT
C##CDB_ADMIN1
37 rows selected.

SQL>
```

Compare Exercising Versus Not Exercising the SYSDBA Privilege

This section compares logging in the C##CDB_ADMIN1 user with and without the SYSDBA privilege.

1. Disconnect from the root container.

```
SQL> DISCONNECT

Disconnected from Oracle Database 12C Enterprise Edition Release 12.2.0.1.0 -
Production
SQL>
```

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

```
SQL> SHOW USER

User is ""
SQL>
```

3. Connect to the root container as C##CDB_ADMIN1 and exercise the SYSDBA privilege. See Appendix - Product-Specific Credentials for the password.

```
SQL> CONNECT C##CDB_ADMIN1/<password> AS SYSDBA

Connected.
SQL>
```

4. Show the current container name.

```
SQL> SHOW CON_NAME

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

5. Show the current user. The current user is SYS, which means the C##CDB_ADMIN1 user can now do anything

that the SYS user can do. Audit trails, however, would still show the c##CDB_ADMIN1 user.

```
SQL> SHOW USER
```

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

- View the list of privileges for the c##CDB_ADMIN1 user by querying the SESSION_PRIVS static data dictionary view. Scroll down to view the privileges listed.

```
SQL> SELECT * FROM session_privs;
```

```
ALTER SYSTEM
AUDIT SYSTEM
CREATE SESSION
...
252 rows selected.
SQL>
```

- Disconnect from the root container.

```
SQL> DISCONNECT
```

```
Disconnected from Oracle Database 12C Enterprise Edition Release 12.2.0.1.0 -
Production
SQL>
```

- Connect to the root container as c##CDB_ADMIN1 again, but this time, do not exercise the SYSDBA privilege. See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CONNECT c##CDB_ADMIN1/<password>
```

```
Connected.
SQL>
```

- Show the current user. You are connected as c##CDB_ADMIN1. Because you included the CONTAINER=ALL clause when granting the CREATE SESSION privilege and DBA role, c##CDB_ADMIN1 can connect as a regular user to any open PDB and perform system and object operations that the DBA role allows.

```
SQL> SHOW USER
```

```
USER is "C##CDB_ADMIN1"
SQL>
```

- View the list of privileges for the c##CDB_ADMIN1 user by querying the SESSION_PRIVS static data dictionary view. Scroll through the list of privileges. Notice that there are fewer privileges listed (239) than when c##CDB_ADMIN1 was connected with the SYSDBA privilege.

```
SQL> SELECT * FROM session_privs;

ALTER SYSTEM
AUDIT SYSTEM
CREATE SESSION
...
239 rows selected.
SQL>
```

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

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

Session altered.
SQL>
```

12. Show the current container. It is PDB1.

```
SQL> SHOW CON_NAME

CON_NAME
-----
PDB1
SQL>
```

Create PDB1_ADMIN

You just connected c##CDB_ADMIN1 to PDB1, which is convenient because you need to be logged into PDB1 to create a local administrator for PDB1. The c##CDB_ADMIN1 user can create the PDB1_ADMIN user.

1. Create a local user named PDB1_ADMIN1 by using the CREATE USER command. See Appendix - Product-Specific Credentials for the password. Assign PDB1_ADMIN1 to the USERS tablespace and the TEMP temporary tablespace. Also unlock the account so that PDB1_ADMIN1 can log in right away. Because this is a local user and not a common user, do not include the CONTAINER=ALL clause.

```
SQL> CREATE USER PDB1_ADMIN1 IDENTIFIED BY <password> DEFAULT TABLESPACE users
      TEMPORARY TABLESPACE temp ACCOUNT UNLOCK;

User created.
SQL>
```

2. Grant PDB1_ADMIN1 the DBA role and the CREATE SESSION privilege in PDB1 only. This is an example of granting a privilege and role locally.

```
SQL> GRANT create session, dba TO PDB1_ADMIN1;

Grant succeeded.
SQL>
```

3. List the local user accounts for PDB1 by querying the DBA_USERS view. The PDB1_ADMIN1 account is

included in the list.

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

USERNAME
-----
PDBADMIN
HR
PDB1_ADMIN1
SQL>
```

- Disconnect c##CDB_ADMIN1 from PDB1.

```
SQL> DISCONNECT

Disconnected from Oracle Database 12C Enterprise Edition Release 12.2.0.1.0 -
64bit Production
SQL>
```

- Connect PDB1_ADMIN1 to PDB1. See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CONNECT PDB1_ADMIN1/<password>@PDB1

Connected.
SQL>
```

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

```
SQL> SHOW USER

USER is "PDB1_ADMIN1"
SQL>
```

- View the list of privileges for PDB1_ADMIN1 by querying the SESSION_PRIVS table. The results below are only some of the privileges returned from the query.

```
SQL> SELECT * FROM session_privs;

ALTER SYSTEM
AUDIT SYSTEM
CREATE SESSION
ALTER SESSION
RESTRICTED SESSION
CREATE TABLESPACE
...
239 rows selected.
SQL>
```

- Try connecting the PDB1_ADMIN1 user to the root container. This user does not have access to the root container, and therefore, you get an error message stating that the user has insufficient privileges. This is the difference between the c##CDB_ADMIN1 user connecting to PDB1 without the SYSDBA privilege. The c##CDB_ADMIN1 user has the DBA role and CREATE SESSION privileges in *all* containers, including the root container. PDB1_ADMIN has the same role and privilege, but only in PDB1.

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;  
ERROR:  
ORA-01031: insufficient privileges  
SQL>
```

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

```
SQL> EXIT  
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64  
bit Production  
$
```

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 5-2 Creating a Local User for an Application

Overview

In this practice, you log in to PDB1 as the local administrator (PDB1_ADMIN1) in SQL*Plus and create a local user account called INVENTORY, which will own the new Inventory software application. INVENTORY is an example of a user account that does not represent a person.

Assumptions

You are currently logged in to VM1 as the oracle user.

Tasks

Create the INVENTORY User Account

1. Open a new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL .

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Start SQL*Plus and connect to PDB1 as the PDB1_ADMIN1 user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus PDB1_ADMIN1/<password>@PDB1

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

3. Create a local user account named INVENTORY. Set the default tablespace to the users tablespace and grant unlimited quota on that tablespace. See Appendix - Product-Specific Credentials for the password.

Note: You can press Return at the end of each line of code. Finish the last line with a semicolon to issue the command. This format makes long SQL statements easier to read.

```
SQL> CREATE USER INVENTORY IDENTIFIED BY <password>
2 DEFAULT TABLESPACE users
3 QUOTA UNLIMITED ON users;

User created.
SQL>
```

4. Grant the CREATE SESSION privilege to INVENTORY.

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

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

```
SQL> SELECT DISTINCT username FROM dba_users WHERE common='NO';
USERNAME
-----
HR
PDBADMIN
PDB1_ADMIN1
INVENTORY
SQL>
```

Connect as INVENTORY and Verify Privileges

1. Disconnect PDB1_ADMIN1 from PDB1.

```
SQL> DISCONNECT

Disconnected from Oracle Database 12C Enterprise Edition Release 12.2.0.1.0 -
Production
SQL>
```

2. Verify that the INVENTORY user account can connect to PDB1. See Appendix - Product-Specific Credentials for the password.

```
SQL> CONNECT INVENTORY/<password>@PDB1

Connected.
SQL>
```

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

```
SQL> SELECT * FROM session_privs;

CREATE SESSION
SQL>
```

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

```
SQL> EXIT  
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64  
bit Production  
$
```

Practice 5-3 Granting the DBA Role to PDBADMIN

Overview

In this practice, you examine the default privileges and roles granted to the PDBADMIN user. PDBADMIN was created for you by DBCA when the CDB and PDB1 were created. This user is intended to operate as the local PDB administrator.

After exploring, you grant PDBADMIN more power with the DBA role so that in later practices, PDBADMIN is able to create profiles, roles, and users.

Assumptions

You are currently logged in to VM1 as the oracle user.

Tasks

Explore the Privileges and Roles Granted to PDBADMIN

1. Open a new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL .

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Start SQL*Plus and connect as the SYS user with the SYSDBA privilege. Note: PDBADMIN does not have the required privileges to view data from the DBA_SYS_PRIVS view in PDB1, which you will do in the next step.

```
$ sqlplus / AS SYSDBA

Connected.
SQL>
```

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

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

4. List the roles granted to the PDBADMIN user by querying the CDB_ROLE_PRIVS view. This view describes the roles granted to all users and roles in the database. The results show that PDBADMIN is granted the PDB_DBA role. Also, the admin option is enabled (ADM=YES), which means that PDBADMIN can grant the PDB_DBA role to other users. The results are formatted for easier viewing.

```
SQL> SELECT granted_role, admin_option FROM cdb_role_privs WHERE
grantee='PDBADMIN';

GRANTED_ROLE ADM
-----
PDB_DBA      YES
SQL>
```

5. List the system privileges granted to the PDB_DBA role by querying the ROLE_SYS_PRIVS view.
 - a. Switch to PDB1. You must be connected to PDB1 to retrieve data, and you must be connected as the SYS user.

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

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

```
SQL> SELECT privilege FROM role_sys_privs WHERE role='PDB_DBA';

PRIVILEGE
-----
CREATE SESSION
SET CONTAINER
CREATE PLUGGABLE DATABASE
SQL>
```
6. List the roles that are granted to the PDB_DBA role by querying the DBA_ROLE_PRIVS view. The results show that the PDB_DBA role is granted the CONNECT role.

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

GRANTED_ROLE
-----
CONNECT
SQL>
```
7. 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';

PRIVILEGE
-----
SET CONTAINER
CREATE SESSION
SQL>
```

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

Grant the DBA Role to PDBADMIN

1. Grant the DBA role locally to PDBADMIN.

```
SQL> GRANT DBA TO PDBADMIN;

Grant succeeded.
SQL>
```

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

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

GRANTED_ROLE
-----
DBA
PDB_DBA
SQL>
```

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

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64
bit Production
$
```

Practice 5-4 Creating a Local Profile in EM Express and Locking Accounts

Overview

In this practice, the PDBADMIN user (local administrator for PDB1) creates a local profile called HPROFILE in Enterprise Manager Database Express (EM Express) to limit the amount of idle time users can have in the PDB. If a user is idle or forgets to log out after 60 minutes, the user session is ended.

Moreover, the profile will automatically lock a database user account if it did not log on after a specified number of days. This locking mechanism is possible with the new 12.2 INACTIVE_ACCOUNT_TIME user resource profile limit.

Tip:

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

To log in to EM Express and perform administrative operations, such as creating profiles, PDBADMIN requires the EM_EXPRESS_ALL role. In the previous practice, you assigned the DBA role to PDBADMIN, which contains the EM_EXPRESS_ALL role.

Assumptions

You are currently logged in to VM1 as the oracle user.

You completed Practice 5-3 Granting the DBA Role to PDBADMIN.

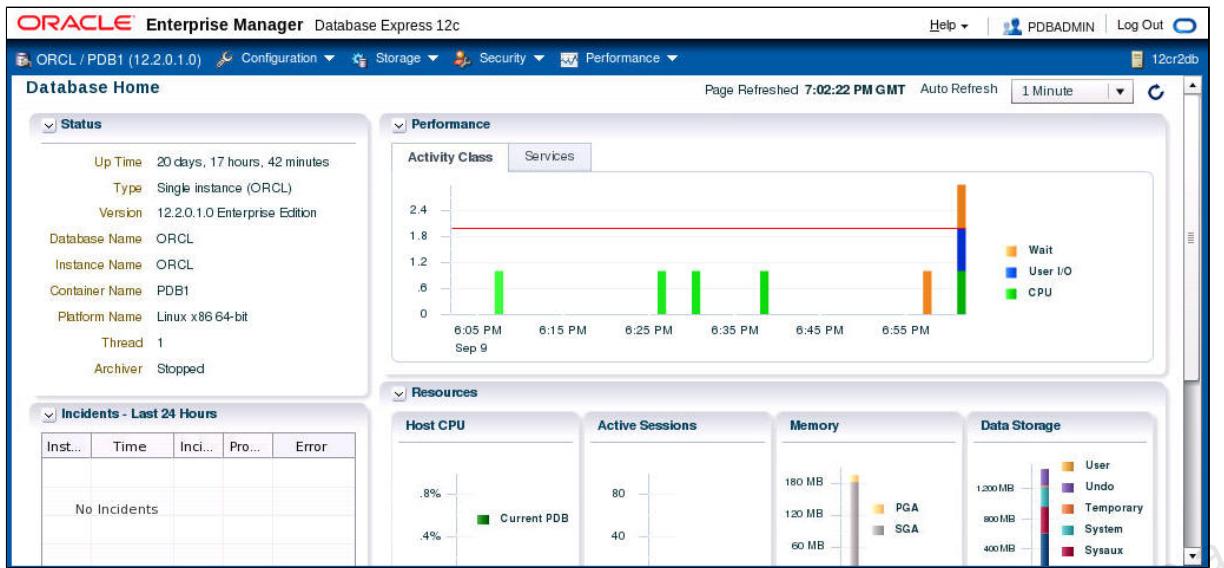
Tasks

Connect the Browser to PDB1 in EM Express

With PDBADMIN given the appropriate privileges, you're ready to log in to EM Express as that user.

1. Open a Firefox browser and enter the URL <https://localhost:5500/em>. Port 5500 is enabled as the global port for PDBs in this CDB.
2. On the Login page for EM Express, log in as PDBADMIN. See [Product-Specific Credentials](#) for the password. Enter PDB1 as the container name. Do not select "as sysdba" check box.
3. Click **Login**.

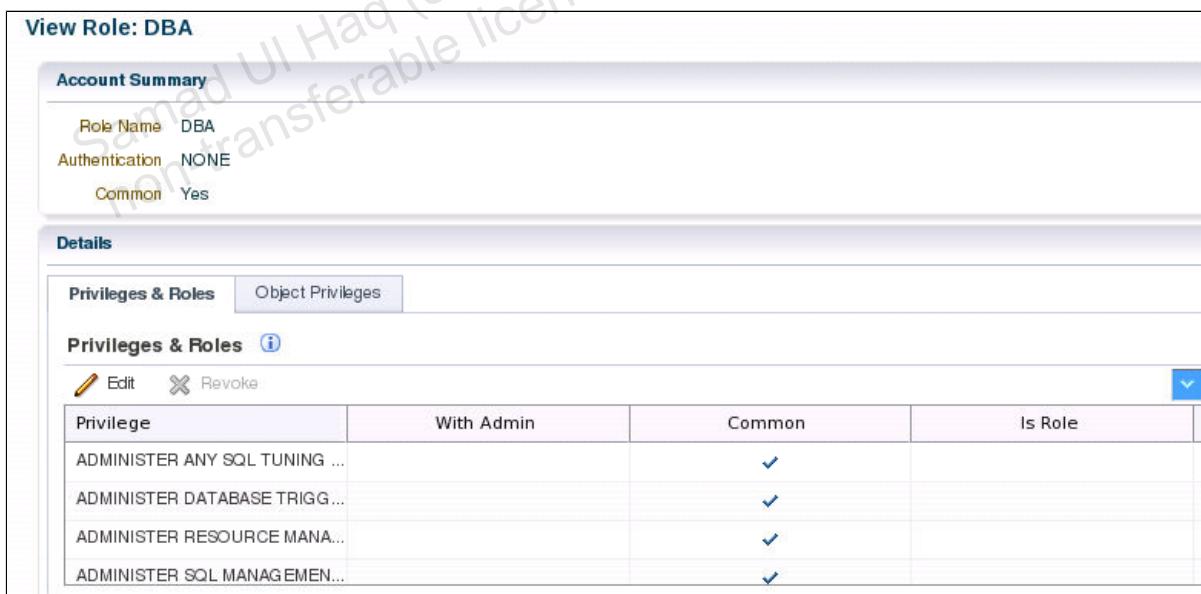
4. View the Database Home page, which reflects information for PDB1. The information on your system will be different than the information in the screenshot below.



Unauthorized reproduction or distribution prohibited. Copyright © 2011 Oracle and/or its affiliates.

[View Privileges and Roles for PDBADMIN](#)

1. On the **Security** menu, select **Users**.
2. Scroll down the list of users, and click **PDBADMIN**. The View User: PDBADMIN page is displayed.
3. On the Privileges and Roles tab, click **DBA**.
4. Scroll down through the list of privileges and roles. Notice that any item in the list that is a role has a check mark in the Is Role column.



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

View Role: DBA

Page Refreshed 9:16:41 PM GMT

Account Summary				
Role Name	DBA	Authentication	NONE	
Common	Yes			

Details

Privileges & Roles		Object Privileges		
Privileges & Roles <small>(i)</small>		<input type="button" value="Edit"/> <input type="button" value="Revoke"/> create profile <input type="button" value="X"/>		
Privilege	With Admin	Common	Is Role	Default Role
CREATE PROFILE		✓		

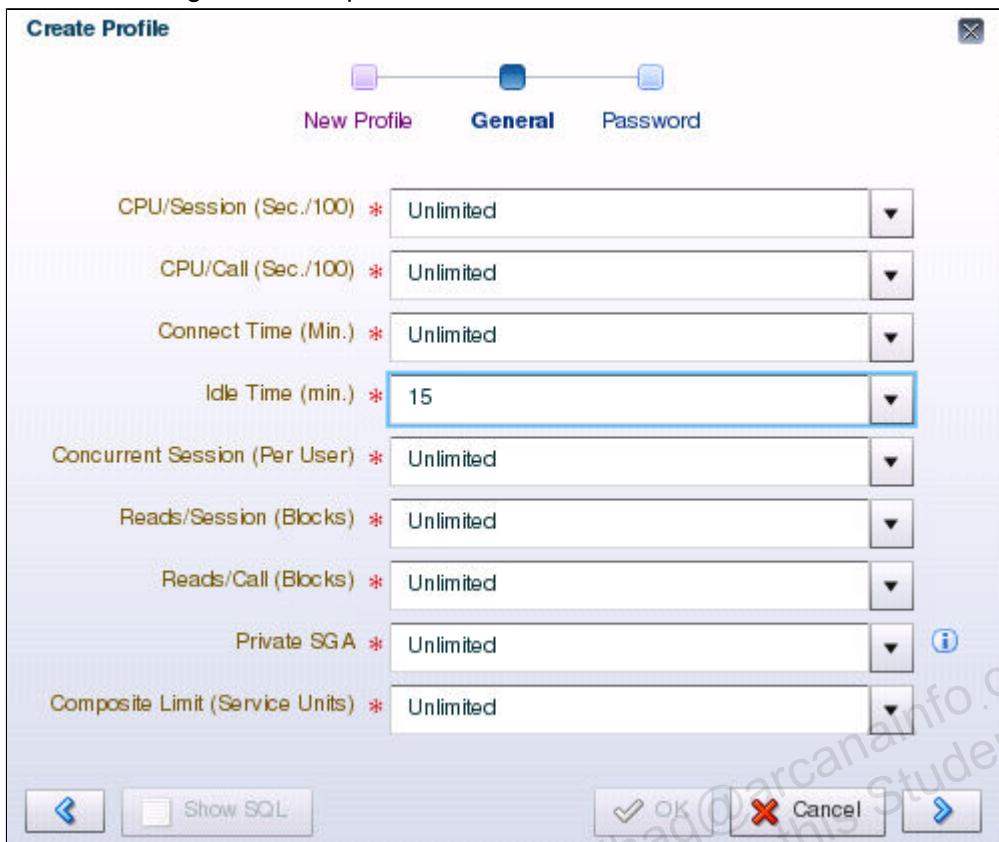
Create a Local Profile

1. On the **Security** menu, select **Profiles**.
2. Click **Create Profile**. The Create Profile wizard is displayed.
3. On the New Profile page, in the Name box, enter **HRPROFILE**, and click **Next**.



4. On the General page, in the Idle Time (Minutes) drop-down list, select **15**. Leave all other fields set to the default value of **Unlimited**. Click **Next**.

Note: You will test the Idle Time setting in Practice 5-6 Creating Local Users in EM Express when you create users and assign them this profile.



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



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

is as follows:

```
create profile 'HRPROFILE' limit
    cpu_per_session UNLIMITED
    cpu_per_call UNLIMITED
    connect_time UNLIMITED
    idle_time 60
    sessions_per_user UNLIMITED
    logical_reads_per_session UNLIMITED

    logical_reads_per_call UNLIMITED
    private_sga UNLIMITED
    composite_limit UNLIMITED
    password_life_time UNLIMITED
...
```

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

Profile	Connect Time (Min.)	Concurrent Session (Per User)
DEFAULT	UNLIMITED	UNLIMITED
HRPROFILE	UNLIMITED	UNLIMITED
ORA_STIG_PROFILE	DEFAULT	DEFAULT

Set the RESOURCE_LIMIT Initialization Parameter

1. On the Configuration menu, select **Initialization Parameters**.
2. In the Name search field, enter **resource_limit**. The RESOURCE_LIMIT initialization parameter is listed in the table.

Name	Value	Modified	Basic	Type	Description
Resource Manager					
resource_limit	true	✓	✓	Boolean	master switch for...

3. Verify that the RESOURCE LIMIT value is set to **true**. If not, perform the following steps:
 - a. Click **Cancel** to return to the table. Select **resource_limit**, and click **Set**.
 - b. In the Set Initialization Parameter dialog box, select the **true** option, and click **OK**.



- c. In the Confirmation dialog box, click **OK**.
4. Click **Logout**, and close the browser window.

Lock Database User Accounts If They Haven't Logged On For 10 Days

To lock database user accounts, modify the `HRRPROFILE` profile to add the `INACTIVE_ACCOUNT_TIME` user resource profile limit. In this section, you use SQL*Plus and learn by trial and error.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Connect to PDB1 as the local DBA.

```
$ sqlplus PDBADMIN/<password>@PDB1

...
SQL>
```

3. Issue the `ALTER PROFILE` command to set the `INACTIVE_ACCOUNT_TIME` limit in the profile to 10 days.

```
SQL> ALTER PROFILE hrprofile LIMIT INACTIVE_ACCOUNT_TIME 10;

ALTER PROFILE hrprofile LIMIT INACTIVE_ACCOUNT_TIME 10
*
ERROR at line 1:
ORA-02377: invalid profile limit INACTIVE_ACCOUNT_TIME
SQL>
```

4. 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.

Answer: The results show a resource named `INACTIVE_ACCOUNT_TIME`, and therefore, yes, `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.

Note: The results below are formatted for easier viewing.

```
SQL> COL limit FORMAT A20
SQL> SELECT resource_type, resource_name, limit FROM dba_profiles WHERE
profile='HRPROFILE';

RESOURCE RESOURCE_NAME LIMIT
-----
KERNEL COMPOSITE_LIMIT UNLIMITED
KERNEL SESSIONS_PER_USER UNLIMITED
KERNEL CPU_PER_SESSION UNLIMITED
KERNEL CPU_PER_CALL UNLIMITED
KERNEL LOGICAL_READS_PER_SESSION UNLIMITED
KERNEL LOGICAL_READS_PER_CALL UNLIMITED
KERNEL IDLE_TIME 15
KERNEL CONNECT_TIME UNLIMITED
KERNEL PRIVATE_SGA UNLIMITED
PASSWORD FAILED_LOGIN_ATTEMPTS UNLIMITED
PASSWORD PASSWORD_LIFE_TIME UNLIMITED
PASSWORD PASSWORD_REUSE_TIME UNLIMITED
PASSWORD PASSWORD_REUSE_MAX UNLIMITED
PASSWORD PASSWORD_VERIFY_FUNCTION NULL
PASSWORD PASSWORD_LOCK_TIME UNLIMITED
PASSWORD PASSWORD_GRACE_TIME UNLIMITED
PASSWORD INACTIVE_ACCOUNT_TIME DEFAULT

17 rows selected.

SQL>
```

- Investigate by displaying the full error message that you received in step 3. To do this, issue the `oerr` command for the error number `ora 2377`. Notice that the error states the limit cannot be less than 15 days.

```
SQL> ! oerr ora 2377

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

SQL>
```

- Set an appropriate limit. Because 10 days is too low, use the lowest valid number instead (which is 15).

```
SQL> ALTER PROFILE hrprofile LIMIT INACTIVE_ACCOUNT_TIME 15;

Profile altered.
```

- Query the `DBA_PROFILES` view again to confirm that the limit is set. Note: The results below are formatted for

easier viewing.

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

RESOURCE RESOURCE_NAME                      LIMIT
----- -----
KERNEL  COMPOSITE_LIMIT                     UNLIMITED
KERNEL  SESSIONS_PER_USER                   UNLIMITED
KERNEL  CPU_PER_SESSION                    UNLIMITED
KERNEL  CPU_PER_CALL                      UNLIMITED
KERNEL  LOGICAL_READS_PER_SESSION          UNLIMITED
KERNEL  LOGICAL_READS_PER_CALL            UNLIMITED
KERNEL  IDLE_TIME                         15
KERNEL  CONNECT_TIME                      UNLIMITED
KERNEL  PRIVATE_SGA                        UNLIMITED
PASSWORD FAILED_LOGIN_ATTEMPTS           UNLIMITED
PASSWORD PASSWORD_LIFE_TIME              UNLIMITED
PASSWORD PASSWORD_REUSE_TIME             UNLIMITED
PASSWORD PASSWORD_REUSE_MAX              UNLIMITED
PASSWORD PASSWORD_VERIFY_FUNCTION        NULL
PASSWORD PASSWORD_LOCK_TIME             UNLIMITED
PASSWORD PASSWORD_GRACE_TIME            UNLIMITED
PASSWORD INACTIVE_ACCOUNT_TIME           15

17 rows selected.

SQL>
```

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

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

9. 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 5-5 Creating Local Roles in EM Express

Overview

In this practice, the PDBADMIN user uses Enterprise Manager Database Express (EM Express) to create the following local roles in PDB1:

- **HRCLERK:** Grant this role the SELECT and UPDATE object privileges on the EMPLOYEES table in the HR schema.
- **HRMANAGER:** Grant this role the SELECT, UPDATE, INSERT, and DELETE object privileges on the entire HR schema.

You will assign these roles to local users in [Practice 5-6 Creating Local Users in EM Express](#).

Assumptions

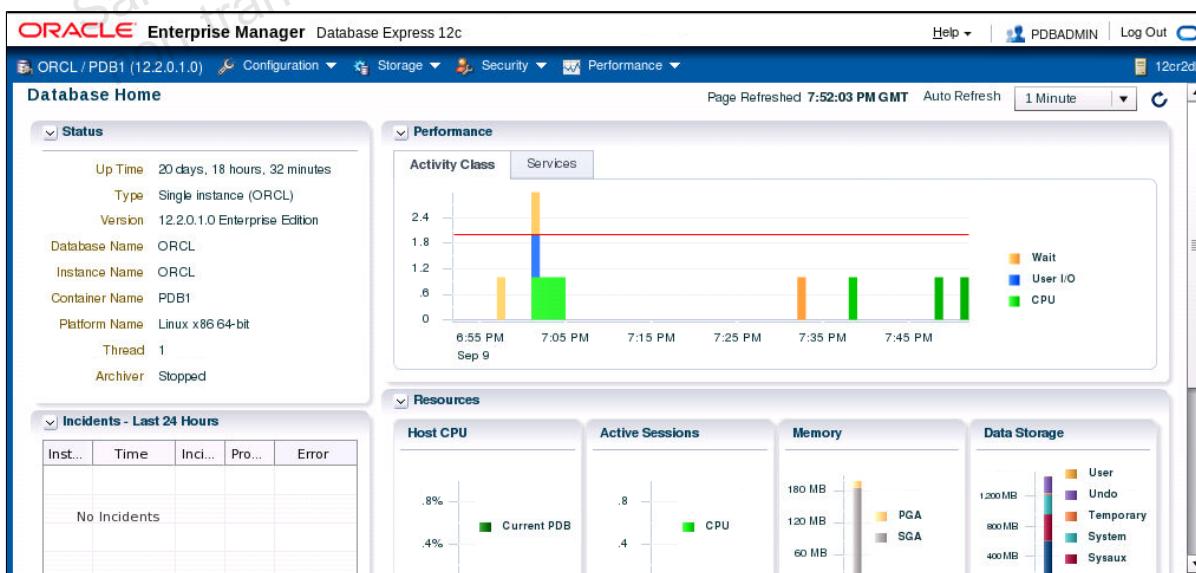
You are currently logged in to VM1 as the oracle user.

You completed [Practice 5-3 Granting the DBA Role to PDBADMIN](#).

Tasks

[Log In to EM Express as PDBADMIN](#)

1. Open a Firefox browser and enter the URL <https://localhost:5500/em> .
2. On the Login page for EM Express, log in as the local database administrator, PDBADMIN. See [Product-Specific Credentials](#) for the password. Enter PDB1 for the container name.
3. Click **Login**.
4. View the Database Home page, which reflects information for PDB1. The information on your system will be different than the information in the screenshot below.



[Create the HRCLERK Role](#)

1. Select **Security**, and then **Roles**. The roles are listed in a table.

Action	Create Role	Drop Role	Role Name
ADM_PARALLEL_EXECUTE_TASK	✓		NONE
APPLICATION_TRACE_VIEWER	✓		NONE
AQ_ADMINISTRATOR_ROLE	✓		NONE
AQ_USER_ROLE	✓		NONE
AUDIT_ADMIN	✓		NONE
AUDIT_VIEWER	✓		NONE
AUTHENTICATEDUSER	✓		NONE
CAPTURE_ADMIN	✓		NONE
CDB_DBA	✓		NONE
CONNECT	✓		NONE
CSW_USR_ROLE	✓		NONE
CTXAPP	✓		NONE
DATADUMP_EVD_FULL_DATABASE			NONE

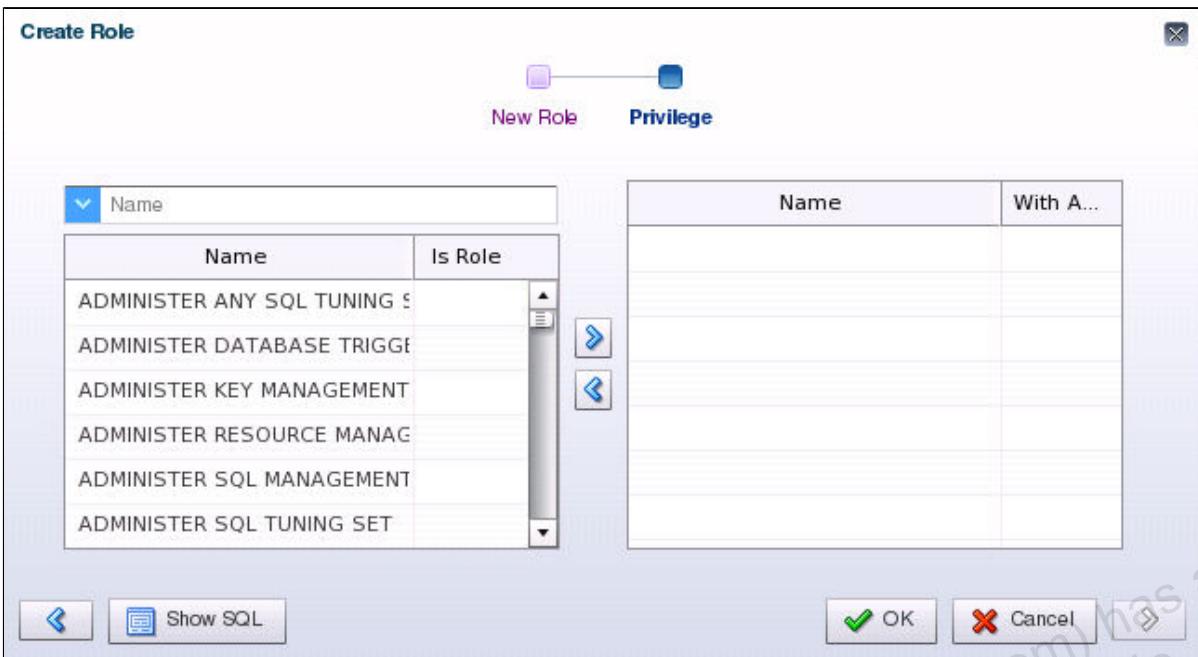
2. Click **Create Role**. The Create Role Wizard is displayed.
 3. On the New Role page, in the Role Name box, enter **HRCLERK**, and click **Next**.
- Create Role**

New Role Privilege

Role Name * **HRCLERK**
4. Click **Show SQL**. The SQL command, as shown below, verifies that the **HRCLERK** role is not yet created. Click **OK**.

```
create role "HRCLERK" NOT IDENTIFIED;
```

5. On the Privilege page, click **OK**. This is not the place where you can assign object privileges for a specific schema.



6. In the Confirmation dialog box, click **OK**.
7. Verify that the HRCLERK role is listed in the table.

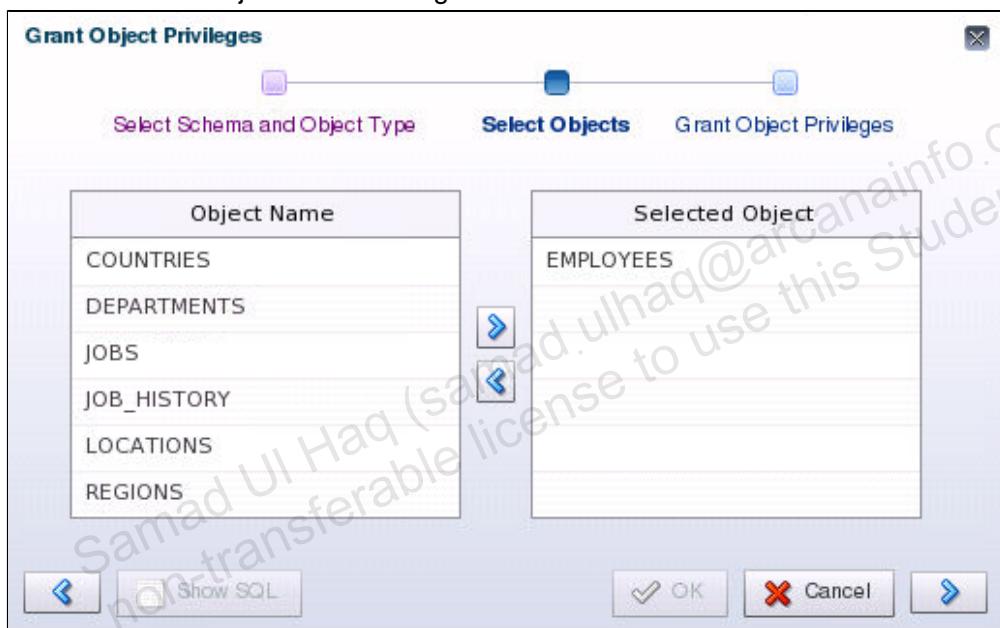
ORACLE® Enterprise Manager Database Express 12c			
ORCL / PDB1 (12.2.0.1.0) Configuration Storage Security Performance			
Roles			
Actions	Create Role	Drop Role	
Role Name	Common	Authentication	
HRCLERK		NONE	
HS_ADMIN_EXECUTE_ROLE	✓	NONE	
HS_ADMIN_ROLE	✓	NONE	
HS_ADMIN_SELECT_ROLE	✓	NONE	
IMP_FULL_DATABASE	✓	NONE	
JAVADEBUGPRIV	✓	NONE	
JAVAIDPRIV		NONE	

8. In the table, select the HRCLERK role (click beside the name). Select **Actions**, and then **Grant Object Privileges**. The Grant Object Privileges Wizard is displayed.
9. On the Select Schema and Object Type page, do the following, and click **Next**.
a. In the Schema drop-down list, select **HR**.
b. In the Object Type drop-down list, leave **TABLE** selected.

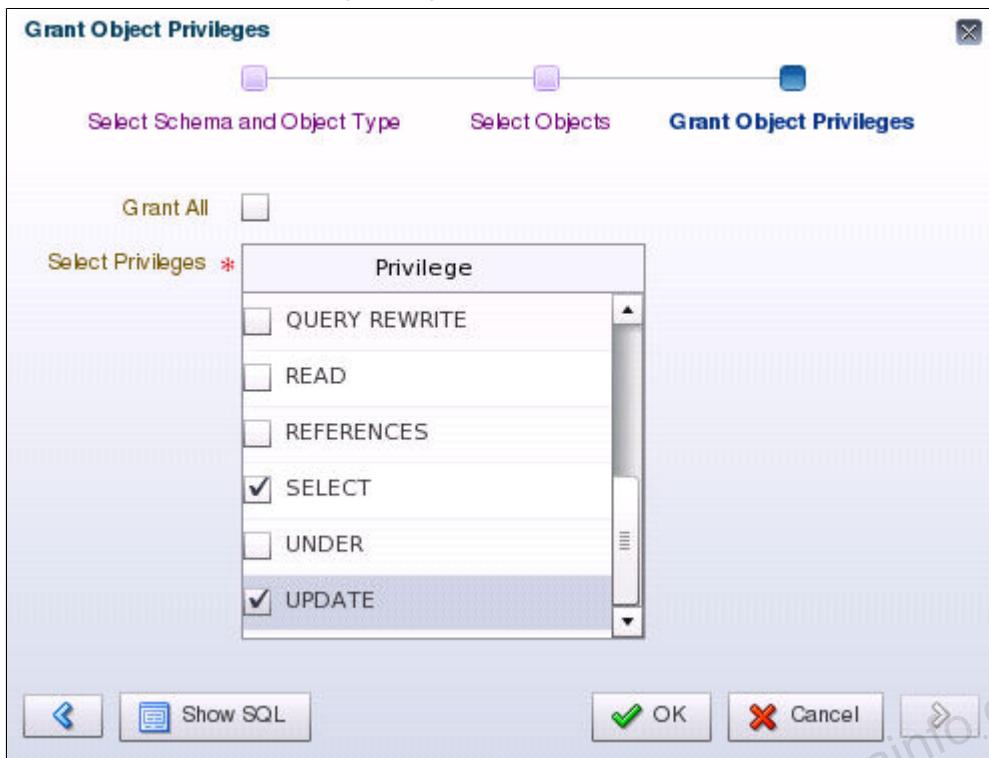
- c. Leave the Object Name box blank.



10. On the Select Objects page, select the EMPLOYEES table on the left, and click the right arrow button to move it to the Selected Object list on the right. Click **Next**.



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



12. Click **Show SQL**. The Confirmation dialog box displays the SQL commands (shown below) that are generated to perform the grant. Click **OK**.

```
grant UPDATE on 'HR'.'EMPLOYEES' to 'HRCLERK';  
grant SELECT on 'HR'.'EMPLOYEES' to 'HRCLERK';
```

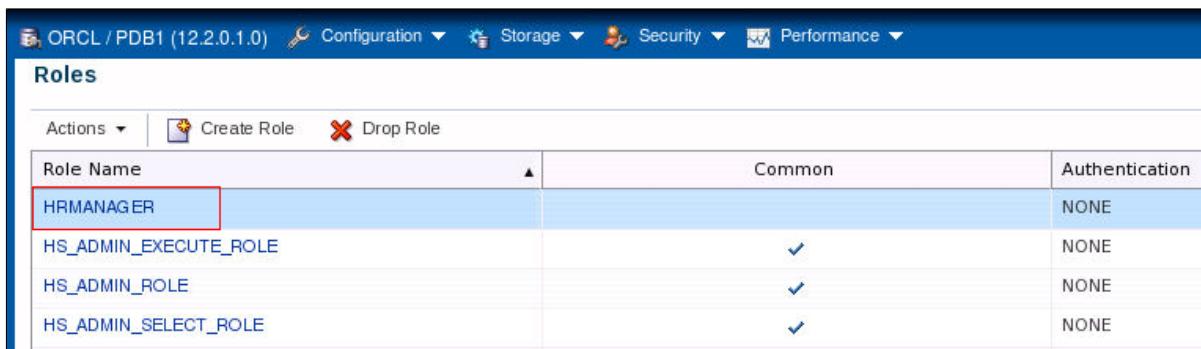
13. On the Grant Object Privileges page, click **OK** to complete the grant.

14. In the Confirmation dialog box, click **OK**. You are finished creating the HRCLERK role.

Create the HRMANAGER Role

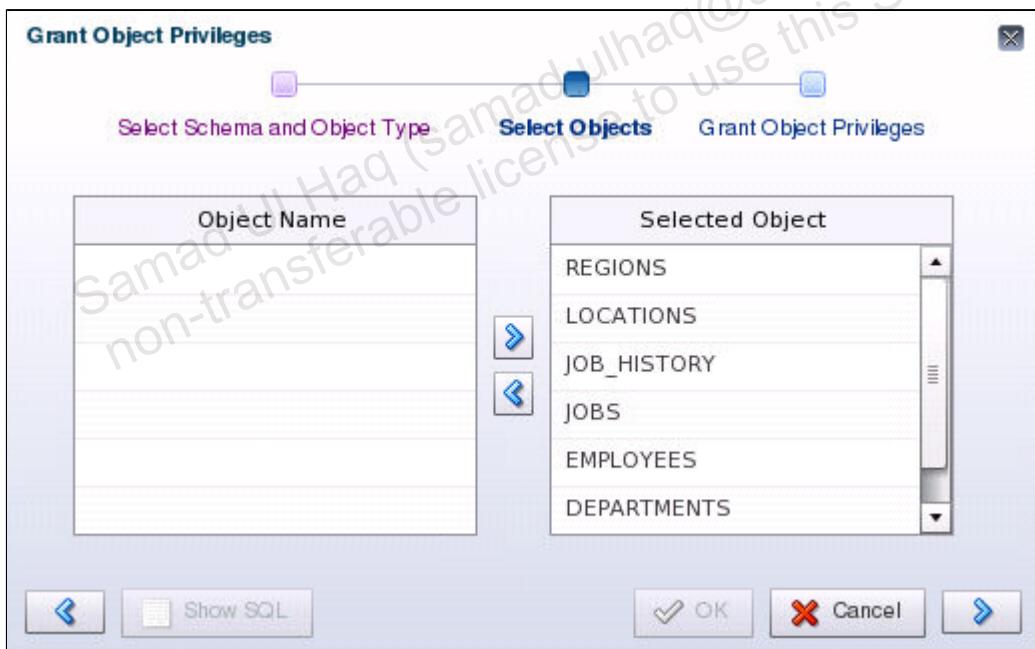
The steps in this section are similar to the ones in the previous section, therefore the screenshots are kept to a minimum.

1. If needed, select **Security**, and then **Roles**. The roles are listed in a table.
2. Click **Create Role**. The Create Role Wizard is displayed.
3. On the New Role page, in the Role Name box, enter **HRMANAGER**, and click **Next**.
4. Click **Show SQL**. The SQL command `create role "HRMANAGER" NOT IDENTIFIED;` verifies that the HRMANAGER role is not yet created. Click **OK**.
5. On the Privilege page, click **OK**. This is not the place where you can assign object privileges for a specific schema.
6. In the Confirmation dialog box, click **OK**.
7. Verify that the HRMANAGER role is listed in the table.

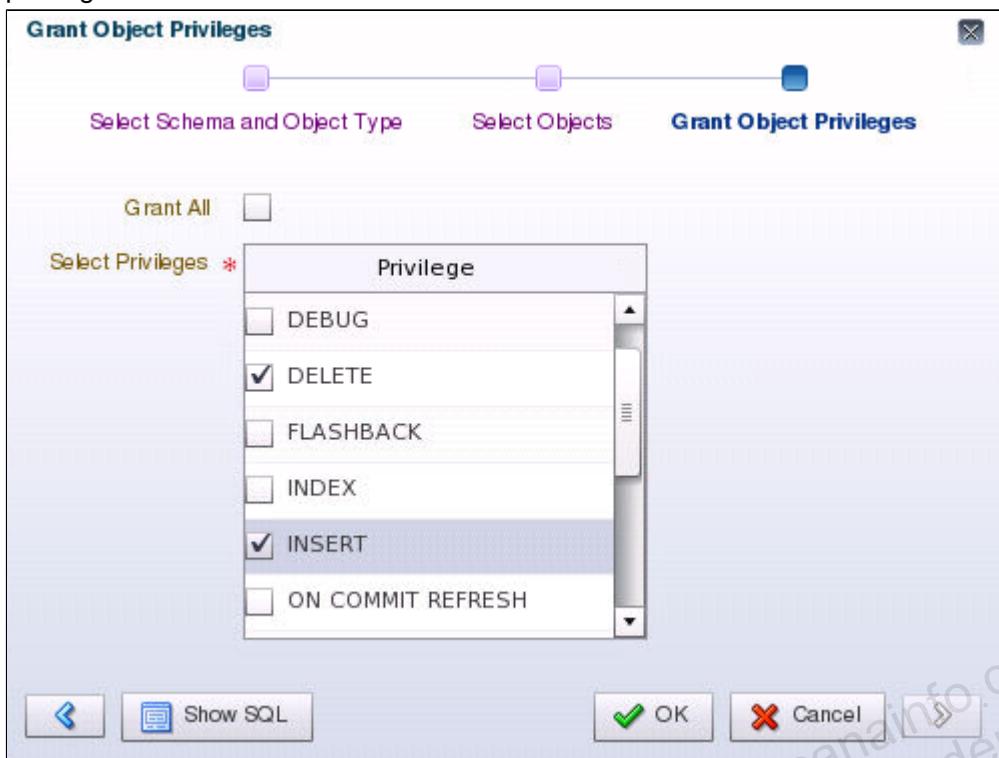


Role Name	Common	Authentication
HRMANAGER		NONE
HS_ADMIN_EXECUTE_ROLE	✓	NONE
HS_ADMIN_ROLE	✓	NONE
HS_ADMIN_SELECT_ROLE	✓	NONE

8. In the table, select the **HRMANAGER** role (click beside the name). Select **Actions**, and then **Grant Object Privileges**. The Grant Object Privileges Wizard is displayed.
9. On the Select Schema and Object Type page, do the following, and click **Next**.
 - a. In the Schema drop-down list, select **HR**.
 - b. In the Object Type drop-down list, select **TABLE**.
 - c. Leave the Object Name box blank.
10. On the Select Objects page, select all the tables on the left (click and then Shift+click), and click the right arrow button to move them to the Selected Object list on the right. The following tables should be selected: REGIONS, LOCATIONS, JOB_HISTORY, JOBS, EMPLOYEES, DEPARTMENTS, and COUNTRIES. Click **Next**.



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



12. Click **Show SQL**. The Confirmation dialog box displays the SQL commands (some of them are shown below) that are generated to perform the grant. Click **OK**.

```
grant UPDATE on 'HR'.'JOBS' to 'HRMANAGER';
grant UPDATE on 'HR'.'COUNTRIES' to 'HRMANAGER';
grant UPDATE on 'HR'.'DEPARTMENTS' to 'HRMANAGER';
grant UPDATE on 'HR'.'JOB_HISTORY' to 'HRMANAGER';
grant UPDATE on 'HR'.'REGIONS' to 'HRMANAGER';
grant UPDATE on 'HR'.'LOCATIONS' to 'HRMANAGER';
grant UPDATE on 'HR'.'EMPLOYEES' to 'HRMANAGER';
grant SELECT on 'HR'.'JOBS' to 'HRMANAGER';
grant SELECT on 'HR'.'COUNTRIES' to 'HRMANAGER';
grant SELECT on 'HR'.'DEPARTMENTS' to 'HRMANAGER';
...

```

13. On the Grant Object Privileges page, click **OK** to complete the grant.
14. In the Confirmation dialog box, click **OK**. You are finished creating the HRMANAGER role.
15. Click **Log Out**, and close the browser window.

Practice 5-6 Creating Local Users in EM Express

Overview

In this practice, you log in to Enterprise Manager Database Express as the PDBADMIN user and create local user accounts in PDB1 according to the following table. Assign the profile named HRPROFILE to the accounts as well as various privileges and roles that you've already created in previous practices. Afterward, test the accounts by logging in to SQL*Plus as each user. Also test the idle time setting in HRPROFILE.

Name	User Account	Description	Privileges/Roles	Method to Create User Account
Jenny Goodman	JGOODMAN	A new HR manager	CREATE SESSION privilege HRCLERK role HRMANAGER role	EM Express interface
David Hamby	DHAMBY	A new Human Resources (HR) clerk	CREATE SESSION privilege HRCLERK role	EM Express interface
Rachel Pandya	RPANDYA	A new HR clerk	CREATE SESSION privilege HRCLERK role	SQL script with substitution variables

Assumptions

You are currently logged in to VM1 as the oracle user.

You created a local profile in PDB1 named HRPROFILE, and two local roles (HRCLERK and HRMANAGER) in PDB1. You also assigned the DBA role to the PDBADMIN user, which is the local administrator for PDB1. If you haven't done so, complete the following practices before starting this one:

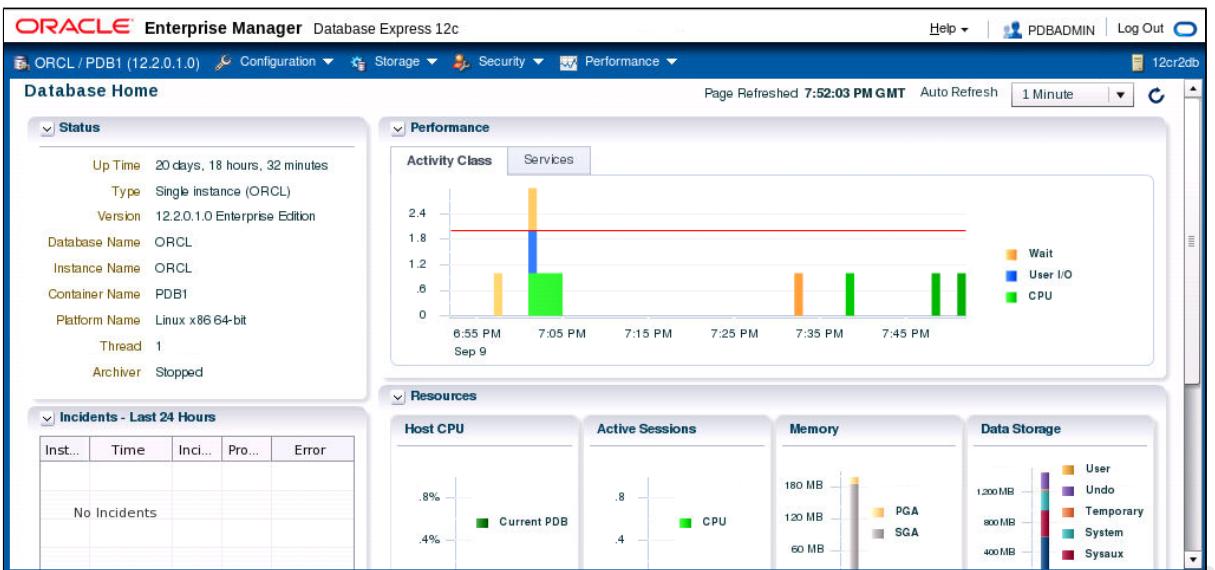
- Practice 5-3 Granting the DBA Role to PDBADMIN
- Practice 5-4 Creating a Local Profile in EM Express and Locking Accounts
- Practice 5-5 Creating Local Roles in EM Express

Tasks

Log In to EM Express as PDBADMIN

1. Open a Firefox browser and enter the URL <https://localhost:5500/em>.
2. On the Login page for EM Express, log in as the local database administrator, PDBADMIN. See [Product-Specific Credentials](#) for the password. Enter PDB1 for the container name.
3. Click **Login**.

4. View the Database Home page, which reflects information for PDB1. The information on your system will be different than the information in the screenshot below.

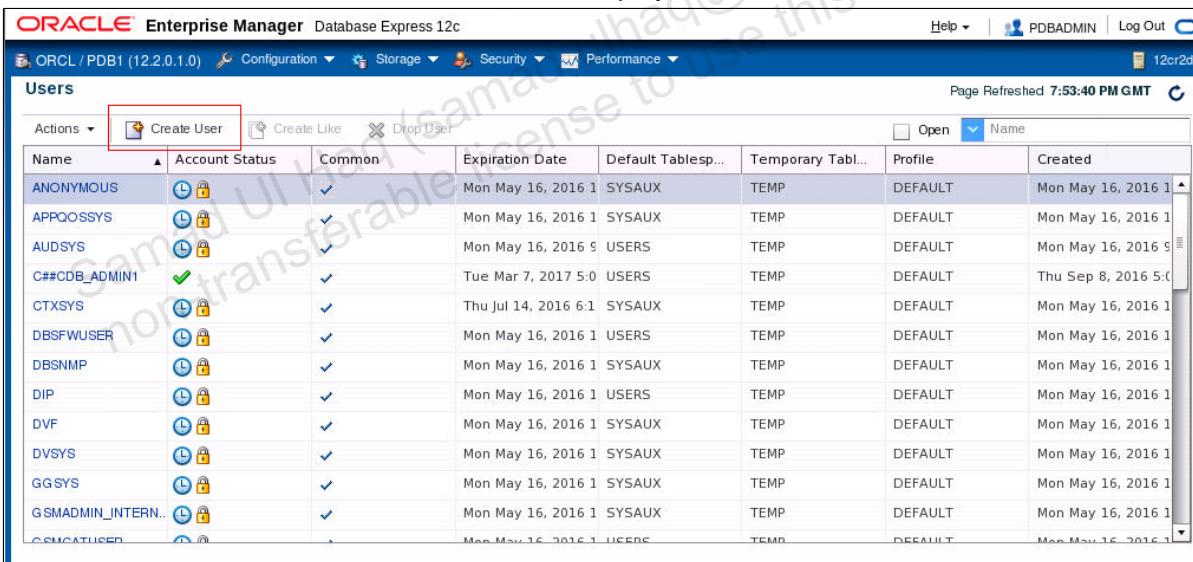


Unauthorized reproduction or distribution prohibited. Copyright © 2017 Oracle and/or its affiliates.

Create a User Account for Jenny Goodman

In this section, you create a user account named JGOODMAN by using the EM Express interface.

1. Select **Security**, and then **Users**.
2. Click **Create User**. The Create User Wizard is displayed.



3. On the User Account page, configure the following, and click **Next**.
 - Name: Enter **JGOODMAN**.
 - Authentication: Leave **Password** selected.
 - Password and Confirm Password: See [Appendix - Product-Specific Credentials](#) for the password.
 - Profile: Select **HRPROFILE**.
 - Leave the check boxes for Password Expired and Account Locked deselected.

Create User

User Account Tablespaces Privilege

Name * JGOODMAN

Authentication * Password External Global

Password * *****

Confirm Password * *****

Profile HRRPROFILE

Password Expired

Account Locked

4. On the Tablespaces page, ensure that the following tablespaces are selected, and click **Next**.

- Default Tablespace: **USERS**
- Temporary Tablespace: **TEMP**

Create User

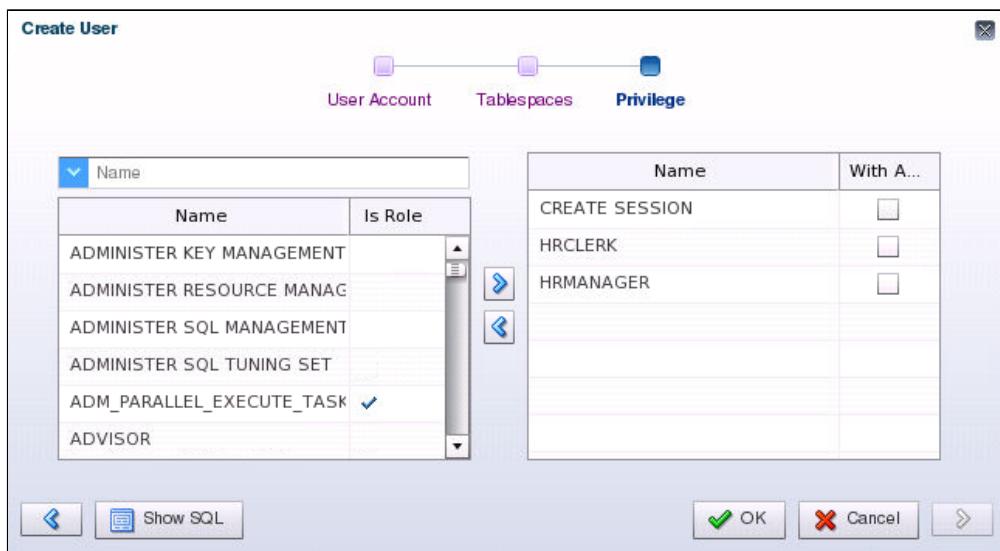
User Account Tablespaces Privilege

Default Tablespace USERS

Temporary Tablespace TEMP

5. On the Privilege page, select the **CREATE SESSION** privilege, **HRCLERK** role, and **HRMANAGER** role, and click the right arrow button to assign them to the user. Click **OK**.

Note: In Practice 5-7 Configuring a Default Role for a User, you will assign the **HRCLERK** role to be this user account's default role.



6. In the Confirmation dialog box, click **OK**.
 7. Verify that **JGOODMAN** is listed in the Users table. From here, you can see that **JGOODMAN** is a local user account (the Common column does not have a check mark), the account is unlocked and the password has not expired (there is no lock or clock in the Account Status column), and the account is assigned the **HRPROFILE** profile.

Name	Account Status	Common	Expiration Date	Default Tablespace	Temporary Tablespace	Profile
JGOODMAN	✓			USERS	TEMP	HRPROFILE
LBACSYS	⌚🔒	▼	Mon May 16, 2016 1	SYSTEM	TEMP	DEFAULT
MDDATA	⌚🔒	▼	Mon May 16, 2016 1	USERS	TEMP	DEFAULT

Create a User Account for David Hamby

In this section, you create another user account named **DHAMBY** by using the EM Express interface. While creating this user, you copy the SQL code to a text file so that in the next section, you can create more users by running a script.

- If needed, select **Security**, and then **Users**.
- Click **Create User**. The Create User Wizard is displayed.
- On the User Account page, configure the following, and click **Next**.
 - Name: Enter **DHAMBY** .
 - Authentication: Leave **Password** selected.
 - Password and Confirm Password: See [Appendix - Product-Specific Credentials](#) for the password.
 - Profile: Select **HRPROFILE** .
 - Select the **Password Expired** check box to force the user to change his password at logon.

Create User

User Account Tablespaces Privilege

Name * DHAMBY

Authentication * Password External Global

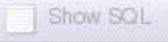
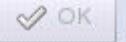
Password * *****

Confirm Password * *****

Profile HRPROFILE

Password Expired 

Account Locked

4. On the Tablespaces page, ensure that the following tablespaces are selected, and click **Next**.

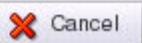
- Default Tablespace: **USERS**
- Temporary Tablespace: **TEMP**

Create User

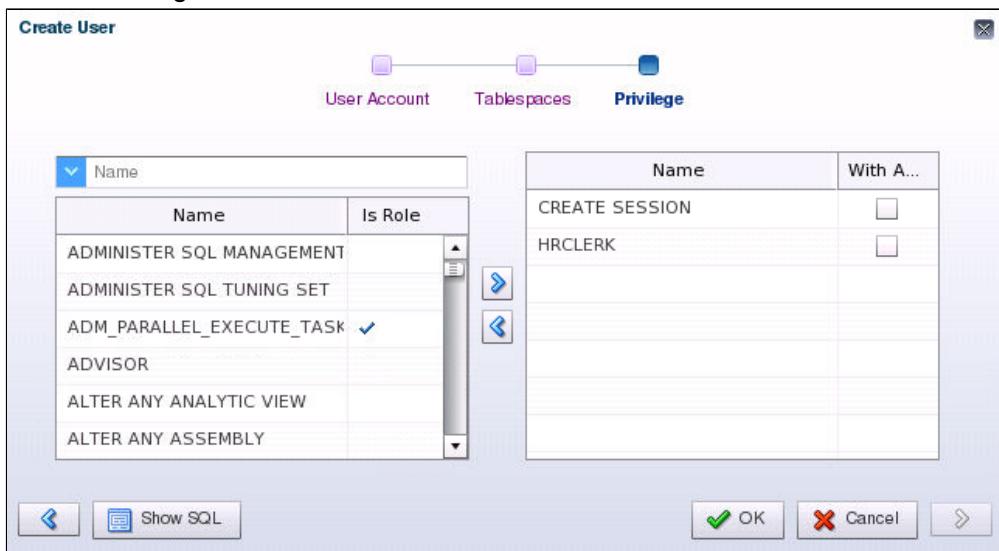
User Account Tablespaces Privilege

Default Tablespace USERS

Temporary Tablespace TEMP

5. On the Privilege page, select the **CREATE SESSION** privilege and **HRCLERK** role, and click the right arrow button to assign them to the user.



6. Click **Show SQL**. A Confirmation dialog box is displayed with the SQL statement (shown below) for the **CREATE USER** and **GRANT** statements. Do not close this dialog box.

```
create user 'DHAMBY' identified by ***** profile 'HRPROFILE' password  
expire account unlock default tablespace 'USERS' temporary tablespace  
  
grant CREATE SESSION to 'DHAMBY';  
grant 'HRCLERK' to 'DHAMBY';
```

7. Create a SQL script that contains the SQL statements displayed in the previous step. Turn the username and role values into substitution variables, rather than hard-coding them. You'll use this script to create future users.

Tip: You can create substitution variables in SQL scripts by using single ampersands (&) and/or double ampersands (&&). A single ampersand indicates to SQL*Plus to prompt you to enter a value each time the substitution variable occurs in the script. A double ampersand indicates to SQL*Plus to prompt you to enter a value only once for a substitution variable, and use that same value for all occurrences of the variable in the script.

- Copy to the clipboard the SQL statements in the previous step.
- Minimize, but don't close, EM Express.
- On the Linux desktop, select **Applications**, then **Accessories**, and then **gedit Text Editor**.
- Select **Edit**, and then **Paste** to paste the SQL statements.
- Change every occurrence of **DHAMBY** to **&&username** . There are three occurrences. This step causes the script to prompt for a username.
- Change the one occurrence of **HRCLERK** to **& &role** . This step causes the script to prompt for a role.
- Replace the asterisks with the actual password. See **Appendix - Product-Specific Credentials** for the password.
- Verify that the code looks like the following code. Don't worry if your **GRANT** statements are in the opposite order. Where you need to make the changes is highlighted in bold.

```

create user "&&username" identified by <password> profile "HRPROFILE"
password expire account unlock default tablespace "USERS" temporary
tablespace "TEMP";
grant CREATE SESSION to "&&username";
grant "&&role" to "&&username";

```

- Unauthorized reproduction of distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.
- i. Select **File**, and then **Save As**. Browse to `/home/oracle`. In the Name box, enter `CreateHRUser.sql`. Click **Save**. The file is saved and formatted.
 - j. Select **File**, and then **Quit**.
 8. Return to EM Express, click **OK** to close the Confirmation dialog box, and click **OK** to finish creating the user.
 9. In the Confirmation dialog box, click **OK**.
 10. Verify that DHAMBY is listed in the Users table. From here, you can see that DHAMBY is a local user account (the Common column does not have a check mark), the account is unlocked and the password has expired (there is no lock in the Account Status column, but there is a clock), and the account is assigned the HRPROFILE profile.

Name	Account Status	Common	Expiration Date	Default Tablesp...	Temporary Tabl...	Profile	Crea
ANONYMOUS	⌚🔒	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon
APQOSSYS	⌚🔓	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon
AUDSYS	⌚🔒	✓	Mon May 16, 2016 9	USERS	TEMP	DEFAULT	Mon
C##CDB_ADMIN1	✓	✓	Tue Mar 7, 2017 5:0	USERS	TEMP	DEFAULT	Thu
CTXSYS	⌚🔒	✓	Thu Jul 14, 2016 6:1	SYSAUX	TEMP	DEFAULT	Mon
DBSFWUSER	⌚🔒	✓	Mon May 16, 2016 1	USERS	TEMP	DEFAULT	Mon
DBSNMP	⌚🔒	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon
DHAMBY	⌚		Fri Sep 9, 2016 8:09	USERS	TEMP	HRPROFILE	Fri Si
DIP	⌚🔒	✓	Mon May 16, 2016 1	USERS	TEMP	DEFAULT	Mon
DVF	⌚🔓	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon
DVSYS	⌚🔒	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon
GGSYS	⌚🔒	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon
SMADMIN_INTCNL	⌚		Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon

Create a User Account for Rachel Pandya by Using a Script

In this section, you create another user account named RPANDYA. Rather than use the EM Express interface to create this user, you use the SQL script that you generated in the previous section.

1. Minimize, but don't close, EM Express.
2. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```

$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$ 

```

3. Start SQL*Plus and connect to PDB1 as the PDBADMIN user. See [Appendix - Product-Specific Credentials for Oracle Database 12c](#) for more information.

the password.

```
$ sqlplus PDBADMIN/<password>@PDB1
```

```
SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.
```

Connected to:

```
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

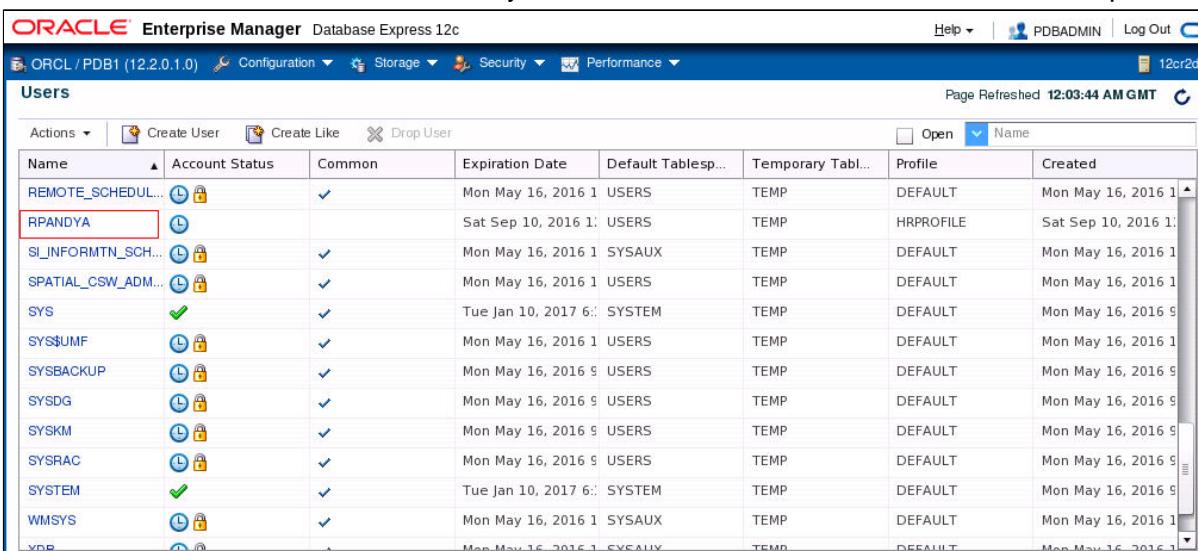
4. Execute the `CreateHRUser.sql` script. Enter `RPANDYA` when prompted for the username. Enter `HRCLERK` when prompted for the role. It is fine if your output is in a slightly different order so long as the `RPANDYA` user is created and granted the `HRCLERK` role and `CREATE SESSION` privilege.

```
SQL> @/home/oracle/CreateHRUser.sql
```

```
Enter value for username: RPANDYA
```

```
old  1: create user "&&username" identified by <password> profile "HRRPROFILE"
password expire account unlock default tablespace  "USERS" temporary tablespace
"TEMP"
new  1: create user "RPANDYA" identified by <password> profile "HRRPROFILE"
password expire account unlock default tablespace  "USERS" temporary tablespace
"TEMP"
User created.
Enter the value for role: HRCLERK
old  1: grant "&&role" to "&&username"
new  1: grant "HRCLERK" to "RPANDYA"
Grant succeeded.
old  1: grant CREATE SESSION to "&&username"
new  1: grant CREATE SESSION to "RPANDYA"
Grant succeeded.
$
```

5. Return to EM Express, and click the **Reload current page** button to refresh the data. Select **Security**, and then **Users**. Scroll down the list and verify that the user RPANDYA has been created as expected.



Name	Account Status	Common	Expiration Date	Default Tablesp...	Temporary Tabl...	Profile	Created
REMOTE_SCHEDUL...	🔒	✓	Mon May 16, 2016 1	USERS	TEMP	DEFAULT	Mon May 16, 2016 1
RPANDYA	🔒	✓	Sat Sep 10, 2016 1:	USERS	TEMP	HRPROFILE	Sat Sep 10, 2016 1:
SI_INFORMTN_SCH...	🔒	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon May 16, 2016 1
SPATIAL_CSW_ADMIN	🔒	✓	Mon May 16, 2016 1	USERS	TEMP	DEFAULT	Mon May 16, 2016 1
SYS	✓	✓	Tue Jan 10, 2017 6:	SYSTEM	TEMP	DEFAULT	Mon May 16, 2016 9
SYS\$UMF	🔒	✓	Mon May 16, 2016 1	USERS	TEMP	DEFAULT	Mon May 16, 2016 1
SYSBACKUP	🔒	✓	Mon May 16, 2016 9	USERS	TEMP	DEFAULT	Mon May 16, 2016 9
SYSDG	🔒	✓	Mon May 16, 2016 9	USERS	TEMP	DEFAULT	Mon May 16, 2016 9
SYSKM	🔒	✓	Mon May 16, 2016 9	USERS	TEMP	DEFAULT	Mon May 16, 2016 9
SYSRAC	🔒	✓	Mon May 16, 2016 9	USERS	TEMP	DEFAULT	Mon May 16, 2016 9
SYSTEM	✓	✓	Tue Jan 10, 2017 6:	SYSTEM	TEMP	DEFAULT	Mon May 16, 2016 9
WMSYS	🔒	✓	Mon May 16, 2016 1	SYSAUX	TEMP	DEFAULT	Mon May 16, 2016 1
vno	⌚	⌚	Mon May 16, 2016 1	SYCAU	TEMP	DEFAULT	Mon May 16, 2016 1

6. Click **Log Out** and close the browser window.

Test DHAMBY's Access in SQL*Plus

Connect to PDB1 as the DHAMBY user. Select the row with employee_id=197 from the HR.EMPLOYEES table. Then attempt to delete it. You should get the “insufficient privileges” error. This happens because in Practice 5-5 Creating Local Roles in EM Express, you granted DHAMBY the HRCLERK role, which has SELECT and UPDATE privileges on the HR.EMPLOYEES table, not INSERT and DELETE.

No need to test for RPANDYA as this user has the same role as DHAMBY.

1. Return to the terminal window.
2. Connect to PDB1 as DHAMBY. When prompted, enter the new password. See [Appendix - Product-Specific Credentials](#) for the original and new password. When you enter the new password, it is not displayed in the interface.

```
SQL> CONNECT DHAMBY/<password>@PDB1

SQL*Plus: Release 12.2.0.1.0 Production on Fri Sep 16 16:01:11 2016

Copyright (c) 1982, 2016, Oracle. All rights reserved.

ERROR:
ORA-28001: the password has expired

Changing password for DHAMBY
New password: <enter new password>
Retype new password: <enter new password>
Password changed

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

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

SALARY.

```
SQL> SELECT salary FROM hr.employees WHERE employee_id=197;
```

```
    SALARY  
-----  
     3000  
SQL>
```

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

```
SQL> DELETE FROM hr.employees WHERE employee_id=197;  
DELETE FROM hr.employees WHERE employee_id=197  
*  
ERROR at line 1:  
ORA-01031: insufficient privileges  
SQL>
```

5. Disconnect from PDB1.

```
SQL> DISCONNECT
```

```
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -  
64bit Production  
SQL>
```

Test JGOODMAN's Access in SQL*Plus

Repeat the test that you just did with DHAMBY, but with the JGOODMAN user account. After deleting the row, issue a rollback, so that you still have the original 107 rows.

1. Connect to PDB1 as JGOODMAN. See [Appendix - Product-Specific Credentials](#) for the password. When creating this user, you did not expire the password, so you won't have to change the password here.

```
SQL> CONNECT JGOODMAN/<password>@PDB1
```

```
Connected.  
SQL>
```

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

```
SQL> SELECT salary FROM hr.employees WHERE employee_id=197;
```

```
    SALARY  
-----  
     3000  
SQL>
```

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

is deleted.

```
SQL> DELETE FROM hr.employees WHERE employee_id=197;  
1 row deleted.  
SQL>
```

4. Roll back the delete operation because this was just a test. You'll learn more about rolling back data later in the course.

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

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

```
SQL> SELECT COUNT(*) FROM hr.employees;  
COUNT(*)  
-----  
107  
SQL>
```

6. Disconnect from PDB1.

```
SQL> DISCONNECT  
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -  
64bit Production  
SQL>
```

Test the Idle Time Limit in HRP PROFILE

If you recall in [Practice 5-4 Creating a Local Profile in EM Express and Locking Accounts](#), you created a profile named HRP PROFILE. In that profile, you configured the Idle Time limit to be 15 minutes. You assigned this profile to all three users (JGOODMAN, DHAMBY, and RPANDYA). In this section, you test that limit by connecting to PDB1 as RPANDYA and letting the session remain inactive for more than 15 minutes. After 15 minutes, verify that RPANDYA was automatically logged out by performing an operation; for example, try to select from the HR.EMPLOYEES table. While you're waiting, you can continue on to the next practice.

1. Connect to PDB1 as RPANDYA. When prompted, enter the new password. See [Appendix - Product-Specific Credentials](#) for the original and new password. When you enter the new password, it is not displayed in the interface.

```
SQL> CONNECT RPANDYA/<password>@PDB1

ERROR:
ORA-28001: the password has expired

Changing password for RPANDYA
New password: <enter new password>
Retype new password: <enter new password>
Password changed
Connected.
SQL>
```

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

```
SQL> SELECT salary FROM hr.employees WHERE employee_id=197;

ERROR at line 1:
ORA-02396: exceeded maximum idle time, please connect again
SQL>
```

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

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -
64bit Production
$
```

Practice 5-7 Configuring a Default Role for a User

Overview

In this practice, PDBADMIN configures HRCLERK as the default role for JGOODMAN (user account for Jenny Goodman in PDB1). Jenny logs into PDB1 with her account and views her privileges that she gets from her default role. She requires more privileges to perform her management tasks, so she enables her non-default role, HRMANAGER, and views her new set of privileges.

Assumptions

You are currently logged in to VM1 as the `oracle` user.

You created the user account called JGOODMAN and granted it the HRMANAGER role, as well as the less-privileged HRCLERK role. To complete this practice, you must first complete the following practices:

- Practice 5-3 Granting the DBA Role to PDBADMIN
- Practice 5-4 Creating a Local Profile in EM Express and Locking Accounts
- Practice 5-5 Creating Local Roles in EM Express
- Practice 5-6 Creating Local Users in EM Express

Tasks

Configure a Default Role for JGOODMAN

1. Open a new terminal window and source the `oraenv` script. For the ORACLE_SID value, enter ORCL.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Start SQL*Plus and connect to PDB1 as the PDBADMIN user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus PDBADMIN/<password>@PDB1

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

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, HRMANAGER and HRCLERK, and both are default roles (the DEF column = YES).

```
SQL> COLUMN granted_role FORMAT A20
SQL> SELECT granted_role, default_role FROM dba_role_privs WHERE
grantee='JGOODMAN';

GRANTED_ROLE          DEF
-----
HRMANAGER            YES
HRCLERK              YES
SQL>
```

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

```
SQL> ALTER USER JGOODMAN DEFAULT ROLE HRCLERK;

User altered.
SQL>
```

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

```
SQL> SELECT granted_role, default_role FROM dba_role_privs WHERE
grantee='JGOODMAN';

GRANTED_ROLE          DEF
-----
HRMANAGER            NO
HRCLERK              YES
SQL>
```

- Disconnect PDBADMIN from PDB1.

```
SQL> DISCONNECT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64
bit Production
SQL>
```

Enable a Non-Default Role

- Connect to PDB1 as JGOODMAN. See Appendix - Product-Specific Credentials for the new password.

```
SQL> CONNECT JGOODMAN/<password>@PDB1

Connected.
SQL>
```

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

```
SQL> SELECT * FROM session_roles;

ROLE
-----
HRCLERK
SQL>
```

3. Suppose JGOODMAN needs to operate as an HR Manager, and not an HR Clerk. Change the default role to be HRMANAGER. Caution: If you use the SET ROLE command, any roles you leave out get disabled.

```
SQL> SET ROLE HRMANAGER;

Role set.
SQL>
```

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

```
SQL> SELECT * FROM session_roles;

ROLE
-----
HRMANAGER
SQL>
```

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

```
SQL> SET ROLE HRMANAGER, HRCLERK;

Role set.
SQL>
```

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

```
SQL> SELECT * FROM session_roles;

ROLE
-----
HRCLERK
HRMANAGER
SQL>
```

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

```
SQL> EXIT

Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64
bit Production
$
```

Practice 5-8 Auditing User Activity

Overview

After having applied the principle of least privilege on your users including the DBAs, you can track changes that those users make or attempt to make in the database.

During auditing operations, you will audit users using specific roles like the predefined DBA role which contains most of the system privileges, granted to special privileged users, and which might be considered for auditing.

Over a period of time, there could be new users with the DBA role granted, like PDBADMIN in PDB1. Some of the earlier DBA users might no longer have the DBA role. The audit-administrator has to keep track of such changing auditing requirements and enable the audit policies appropriately to new sets of DBA users. Similarly, users who no longer have the DBA role granted should be excluded from auditing to avoid generating unnecessary audit records. This is a tedious and repetitive admin task. The 12.2 enhancement allows you to enable an audit policy on the DBA role so that it is effective for all users to whom the DBA role is granted.

Assumptions

You are currently logged in to VM1 as the `oracle` user.

You completed [Practice 5-3 Granting the DBA Role to PDBADMIN](#).

Tasks

Enable Unified Auditing

Complete the following steps on VM1.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. In PDB1, execute the `$HOME/labs/enable_unified_auditingv2.sh` shell script to enable Unified Auditing. This script stops the listener, shuts down the database instance, relinks Oracle with the `uniaud_on` option, and then restarts the database instance and listener. This script takes about one minute to run.

```
$ $HOME/labs/enable_unified_auditingv2.sh

...
$
```

3. Start SQL*Plus and connect to PDB1 as the `SYSTEM` user.

```
$ sqlplus system/<password>@PDB1

...
SQL>
```

- Verify that unified auditing is now enabled. The results are formatted for easier viewing.

```
SQL> SELECT * FROM v$option WHERE parameter = 'Unified Auditing';

PARAMETER          VALUE  CON_ID
-----
Unified Auditing    TRUE      0
SQL>
```

Create and Enable an Audit Policy

- Create an audit policy to control users who drop or attempt to drop application tables via the DBA role.

```
SQL> CREATE AUDIT POLICY drop_pol PRIVILEGES DROP ANY TABLE;
Audit policy created.
SQL>
```

- Enable the audit policy for users granted the DBA role.

```
SQL> AUDIT POLICY drop_pol BY USERS WITH GRANTED ROLES dba;
Audit succeeded.
SQL>
```

- Verify that the audit policy is enabled for users granted the DBA role.

```
SQL> SELECT entity_name, entity_type, enabled_option
  2  FROM  audit_unified_enabled_policies
  3 WHERE policy_name = 'DROP_POL';

ENTITY_NAME ENTITY_  ENABLED_OPTION
-----
DBA        ROLE      BY GRANTED ROLE
SQL>
```

Test the Audit Policy with the PDBADMIN User

Connect to PDB1 as the PDBADMIN user and create and drop a table. Then, verify that the attempt to drop a table was audited.

- Connect to PDB1 as PDBADMIN.

```
SQL> CONNECT PDBADMIN/<password>@PDB1
```

Connected.
SQL>

2. Create a table in the HR schema named TEST (HR.TEST).

```
SQL> CREATE TABLE hr.test (c NUMBER);
```

Table created.
SQL>

3. Drop the HR.TEST table. This is the activity that you are auditing.

```
SQL> DROP TABLE hr.test;
```

Table dropped.
SQL>

4. Connect to PDB1 as the SYSTEM user.

```
SQL> CONNECT SYSTEM/<password>@PDB1
```

Connected.
SQL>

5. View the audit information. The results show that the DROP TABLE action was audited.

```
SQL> SELECT dbusername, action_name, object_name
  2  FROM unified_audit_trail
  3 WHERE dbusername = 'PDBADMIN' AND action_name = 'DROP TABLE';

DBUSERNAME ACTION_NAME          OBJECT_NAM
----- -----
PDBADMIN   DROP TABLE           TEST
SQL>
```

Test the Audit Policy with a Junior DBA

Create a junior DBA and grant him the DBA role. Then, check if the new junior DBA is automatically audited when he attempts to drop a table. You should be currently logged in to PDB1 as the SYSTEM user.

1. Create the junior DBA and grant him the DBA role. See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CREATE USER john IDENTIFIED BY <password>;
```

User created.
SQL>

2. Grant john the DBA role.

```
SQL> GRANT create session, DBA to john;
```

Grant succeeded.
SQL>

3. Create a table named list.

```
SQL> CREATE TABLE list (user_name VARCHAR2(20));
```

Table created.
SQL>

4. Connect to PDB1 as john and drop the list table. See [Appendix - Product-Specific Credentials](#) for the password.

- a. Connect to PDB1.

```
SQL> CONNECT john/<password>@PDB1
```

Connected.
SQL>

- b. Drop the list table. This is an important table that was created by the SYSTEM user for his own purpose.

```
SQL> DROP TABLE system.list;
```

Table dropped.
SQL>

5. Connect to PDB1 as the SYSTEM user and verify that john's action was audited. See [Appendix - Product-Specific Credentials](#) for the password.

- a. Connect to PDB1.

```
SQL> CONNECT SYSTEM/<password>@PDB1
```

Connected.
SQL>

- b. Execute the following query. The results indicate that john's action was audited. Even though you didn't add john to the list of audited users, he was still audited because he had been granted the DBA role.

```
SQL> SELECT dbusername, action_name, object_name
  2  FROM unified_audit_trail
  3  WHERE dbusername = 'JOHN';

DBUSERNAME ACTION_NAME OBJECT_NAM
-----
JOHN      DROP TABLE LIST
SQL>
```

Drop the Audit Policy

1. Try to drop the audit policy. You are currently connected to PDB1 as the SYSTEM user. You receive an error stating the audit policy cannot be dropped because it is currently enabled. This message indicates that you need to disable the audit policy before you can drop it.

```
SQL> DROP AUDIT POLICY drop_pol;

DROP AUDIT POLICY drop_pol
*
ERROR at line 1:
ORA-46361: Audit policy cannot be dropped as it is currently
enabled.
SQL>
```

2. Disable the audit policy.

```
SQL> NOAUDIT POLICY drop_pol;

Noaudit succeeded.
SQL>
```

3. Drop the audit policy.

```
SQL> DROP AUDIT POLICY drop_pol;

DROP AUDIT POLICY drop_pol
*
ERROR at line 1:
ORA-46361: Audit policy cannot be dropped as it is currently
enabled.
SQL>
```

4. Question: Why can't you drop the audit policy even though you disabled it?

Answer: The answer lies in how you created the audit policy. Did you use options? If so, you must include those same options when disabling the policy.

5. Disable the audit policy again, but this time, include the option BY USERS WITH GRANTED ROLES dba.

```
SQL> NOAUDIT POLICY drop_pol BY USERS WITH GRANTED ROLES dba;  
Noaudit succeeded.  
SQL>
```

6. Drop the audit policy. The results indicate that the policy was dropped successfully.

```
SQL> DROP AUDIT POLICY drop_pol;  
Audit Policy dropped.  
SQL>
```

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

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

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Practice 5-9 Exploring OS and Password File Authentication

Overview

In this practice, you explore the configuration for OS and password file authentication on VM1.

Assumptions

You are currently logged in to VM1 as the `oracle` user.

Tasks

Exploring OS Authentication

Many times during the course, you have logged on to the Oracle database as the `oracle` user and were authenticated using OS authentication. This section explores the groups and users in the Linux OS, and how they are linked to authentication.

1. Open a new terminal window.
2. 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 VM1. The format of each line is `group_name:password:Group ID (GID):user_list`. The Oracle Universal Installer creates the `oinstall` and `dba` groups in the OS. Notice that these groups are included in the list below. The `dba` group consists of the `oracle` user. The `oinstall` group does not have any users listed.

```
$ cat /etc/group

root:x:0:
bin:x:1:bin,daemon
daemon:x:2:bin,daemon
sys:x:3:bin,adm
adm:x:4:adm,daemon
tty:x:5:
disk:x:6:
lp:x:7:daemon
mem:x:8:
kmem:x:9:
wheel:x:10:
mail:x:12:mail,postfix
uucp:x:14:
man:x:15:
games:x:20:
gopher:x:30:
video:x:39:
dip:x:40:
ftp:x:50:
lock:x:54:
audio:x:63:
nobody:x:99:
users:x:100:
dbus:x:81:
```

```
utmp:x:22:  
utempter:x:35:  
floppy:x:19:  
vcsa:x:69:  
rpc:x:32:  
cdrom:x:11:  
tape:x:33:  
dialout:x:18:  
wbpriv:x:88:  
ntp:x:38:  
saslauth:x:76:  
postdrop:x:90:  
postfix:x:89:  
rpcuser:x:29:  
nfsnobody:x:65534:  
abrt:x:173:  
cgred:x:499:  
haldaemon:x:68:haldaemon  
stapusr:x:156:  
stapsys:x:157:  
stapdev:x:158:  
sshd:x:74:  
tcpdump:x:72:  
slocate:x:21:  
oprofile:x:16:  
desktop_admin_r:x:498:  
desktop_user_r:x:497:  
fuse:x:496:  
ctapiusers:x:495:  
rtkit:x:494:  
pulse:x:493:  
pulse-access:x:492:  
gdm:x:42:  
oinstall:x:54321:  
dba:x:54322:oracle  
apache:x:48:  
avahi:x:70:  
avahi-autoipd:x:170:  
dhcpd:x:177:  
named:x:25:  
vncuser:x:54323:  
hsqldb:x:96:  
nx:x:491:  
$
```

3. You can find help information for the group file by executing the following command. Enter **q** to exit.

```
$ man GROUP
GROUP(5)           Linux Programmer's Manual          GROUP(5)

NAME
group - user group file

DESCRIPTION
/etc/group is a text file which defines the groups on the system.
There is one entry per line, with the following format:

group_name:passwd:GID:user_list

The field descriptions are:

group_name
the name of the group.

password
the (encrypted) group password. If this field is empty, no
password is needed.

GID      the numerical group ID.

user_list
:q
$
```

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

```
$ whoami
oracle
$
```

5. Find out more about the `oracle` user, for example, prove that `oracle` is part of the `dba` group.

- a. The `/etc/passwd` file is a text file that lists user account information needed for logging in to the OS. Execute the following command to search for the `oracle` user. The format of the row is `user:password:user ID:primary group ID:home directory:shell that would run`. Passwords are stored in the `/etc/shadow` file, so an `x` is used here instead.

```
$ grep oracle /etc/passwd
oracle:x:54321:54321::/home/oracle:/bin/bash
$
```

- b. The information above tells you that `oracle`'s primary group ID is 54321. To find the name of that group, search for it in the group file. The result shows that the `oracle` user's primary group is the `oinstall` group. However, a few steps back you saw that the `oinstall` group didn't have any users listed. This means that `oracle` is a user within a subgroup of `oinstall`.

```
$ grep 54321 /etc/group
oinstall:x:54321
$
```

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

```
$ grep oracle /etc/group
dba:x:54322:oracle
$
```

Exploring Password Authentication

When you grant an administrative privilege to a user, for example, `SYSDBA` or `SYSOPER`, that user's name and privilege information are added to the database password file. The `V$PFILE_USERS` view contains information about users that have been granted administrative privileges.

1. Source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

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

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

```
$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

3. View the columns in the `V$PFILE_USERS` view by issuing the `DESCRIBE` command.

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

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

4. List the users in the password file by querying the V\$PFILE_USERS view.

```
SQL> SELECT username FROM v$pwfile_users;
USERNAME
-----
SYS
SYSDG
SYSBACKUP
SYSKM
C##CDB_ADMIN1
SQL>
```

5. 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 cannot connect anymore), or EXPIRED (the user is mandated to change the password at the connection).

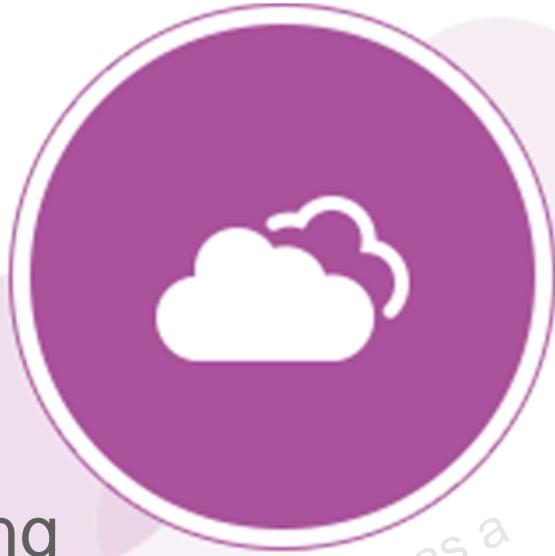
```
SQL> SELECT account_status, sysdba from v$pwfile_users WHERE username='SYS';
ACCOUNT_STATUS          SYSDBA
OPEN                      TRUE
SQL>
```

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

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

6

Creating and Managing Tablespaces



ORACLE®

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Objectives for Lesson 6

After completing this lesson, you should be able to:

- Explain how table data is stored in the database
- Create tablespaces in SQL*Plus
- Alter or drop tablespaces in SQL*Plus
- View tablespace information in SQL*Plus
- Implement Oracle Managed Files (OMF)
- Move or rename online data files in SQL*Plus

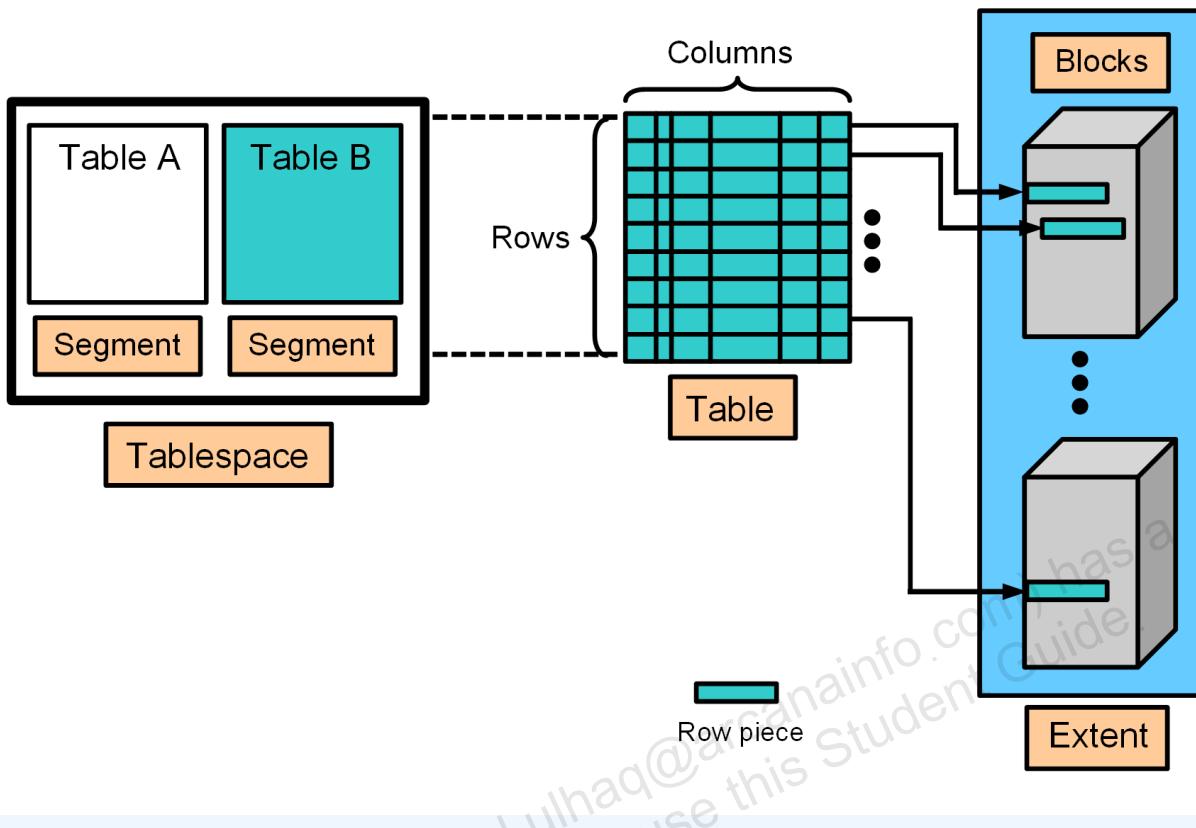


ORACLE

The size of a PDB can be described as the sum of all of its tablespaces. As you manage your PDBs over time, you may need to enlarge them as usage increases. You can do that by creating new tablespaces, adding data files to existing smallfile tablespaces, increasing the size of data files, and providing for the dynamic growth of your data files. All these activities can be performed with SQL statements, although if you prefer working with a GUI, you can use Enterprise Manager Cloud Control or Enterprise Manager Database Express (EM Express).

Before starting, recall in Lesson 1 when you learned about **default tablespaces** (SYSAUX, SYSTEM, TEMP, UNDO, and USERS), which get created automatically in CDBs and PDBs. It would be helpful to review those details at this time, if needed, before starting this lesson.

How Table Data Is Stored

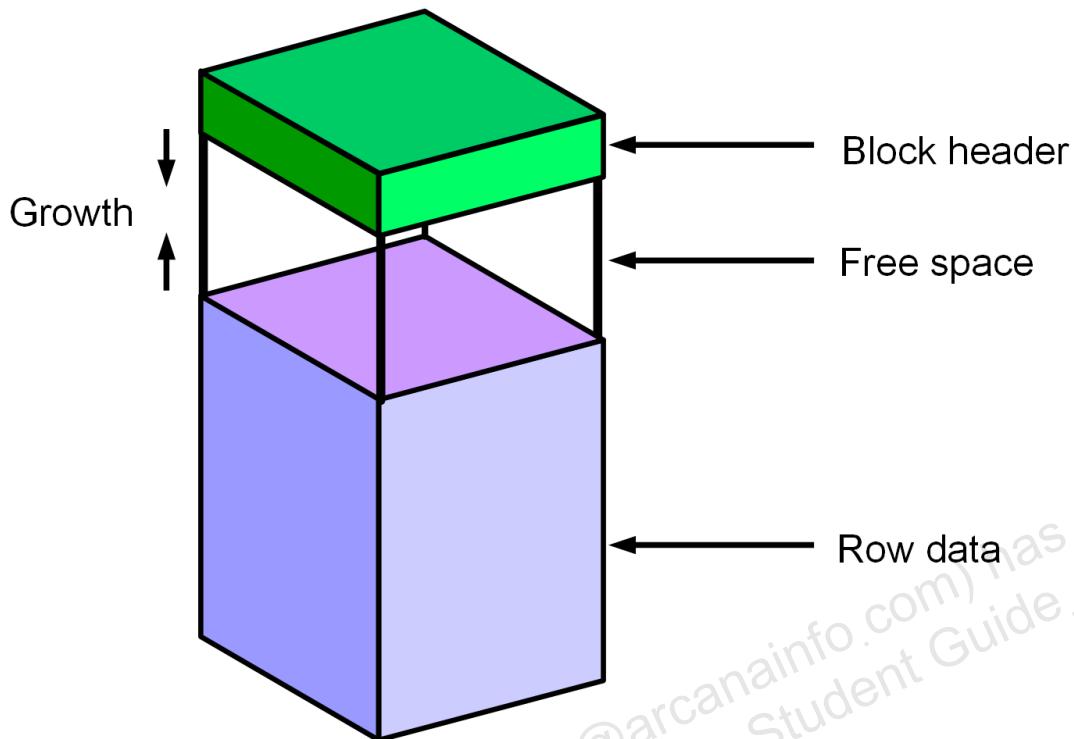


ORACLE®

When a table is created, a logical "segment" is created to hold its data, as illustrated above. A tablespace contains a collection of segments.

Logically, a table contains rows of column values. A row is ultimately stored in a database block in the form of a row piece (also illustrated above). It is called a row piece because, under some circumstances, the entire row may not be stored in one place. This happens when an inserted row is too large to fit into a single block (chained row) or when an update causes an existing row to outgrow the available free space of the current block (migrated row). Row pieces are also used when a table has more than 255 columns. In this case the pieces may be in the same block (intra-block chaining) or across multiple blocks.

Database Block Content



ORACLE

A database block contains a block header, row data, and free space, as illustrated above.

- A block header contains the segment type (such as table or index), data block address, table directory, row directory, and transaction slots of approximately 23 bytes each, which are used when modifications are made to rows in the block. The block header grows downward from the top.
- Row data is the actual data for the rows in the block. Row data space grows upward from the bottom.
- Free space is in the middle of the block, enabling the header and the row data space to grow when necessary. Row data takes up free space as new rows are inserted or as columns of existing rows are updated with larger values. Examples of events that cause header growth:
 - Row directories that need more row entries
 - More transaction slots required than initially configured

Initially, the free space in a block is contiguous. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the Oracle server when necessary.

Creating Tablespaces

- A tablespace is an allocation of space in the database that can contain schema objects.
- Create a tablespace with the `CREATE TABLESPACE` statement or a graphical interface, such as EM Express.
- You can create three types of tablespaces:
 - **Permanent tablespace:** Contains persistent schema objects. Objects in permanent tablespaces are stored in data files.
 - **Undo tablespace:** Is a type of permanent tablespace used by Oracle Database to manage undo data if you are running your database in automatic undo management mode.
 - Oracle strongly recommends that you use automatic undo management mode rather than using rollback segments for undo.
 - **Temporary tablespace:** Contains schema objects only for the duration of a session.
 - Objects in temporary tablespaces are stored in temp files.

ORACLE®

Prerequisites for CREATE TABLESPACE

Before you can create a tablespace, you must create a database to contain it, and the database must be open. You must also have the `CREATE TABLESPACE` system privilege. To create the `SYSAUX` tablespace, you must have the `SYSDBA` system privilege.

File Name and Size

You must include the `DATAFILE` or `TEMPFILE` clause when you create a tablespace. Use this clause to specify the name and location of the data file or temp file. A tablespace must have at least one data file or temp file. You must also specify an initial file size.

You can include the `AUTOEXTEND ON` clause to automatically extend the file when it is full. In this case, you'll need to specify an increment amount and a maximum file size, which can be unlimited. Remember, the size of the file is limited by the physical media on which it resides.

You can include the `BIGFILE` or `SMALLFILE` clause to override the default tablespace type (permanent or temporary) for the database. If you omit this clause, then Oracle Database uses the current default tablespace type.

- A *bigfile tablespace* contains only one data file (or temp file), which can contain up to approximately 4 billion blocks. Bigfile tablespaces are used with extremely large databases, in which Automatic Storage Management

(ASM) or other logical volume managers support the striping or redundant array of independent disks (RAID) and dynamically extensible logical volumes. For bigfile tablespaces, you can specify only one data file in the DATAFILE clause or one temp file in the TEMPFILE clause.

- A *smallfile tablespace* is a traditional Oracle tablespace, which can contain 1022 data files or temp files, each of which can contain up to approximately 4 million blocks.

Availability

You can include the ONLINE or OFFLINE clause to make the tablespace available (or not available) immediately after creation to users who have been granted access to the tablespace. ONLINE is the default. The data dictionary view DBA_TABLESPACES indicates whether each tablespace is online or offline. This clause cannot be used with temporary tablespaces.

Block Size

You can include the BLOCKSIZE clause to specify a nonstandard block size. In order to specify this clause, the DB_CACHE_SIZE and at least one DB_n_K_CACHE_SIZE parameter must be set, and the integer you specify in this clause must correspond with the setting of one DB_n_K_CACHE_SIZE parameter setting. You cannot specify nonstandard block sizes for a temporary tablespace or if you intend to assign this tablespace as the temporary tablespace for any users. If you don't specify a block size, the database will use the default 8KB block size for the tablespace.

Extent Management

You can include the EXTENT MANAGEMENT clause to specify how the extents of the tablespace will be managed.

- AUTOALLOCATE specifies that the tablespace is system managed. Users cannot specify an extent size. You cannot specify AUTOALLOCATE for a temporary tablespace.
- UNIFORM value specifies that the tablespace is managed with uniform extents of SIZE bytes. The default SIZE is 1MB. All extents of temporary tablespaces are of uniform size, so this keyword is optional for a temporary tablespace. However, you must specify UNIFORM in order to specify SIZE. You cannot specify UNIFORM for an undo tablespace.

If you don't specify AUTOALLOCATE or UNIFORM, then the default is UNIFORM for temporary tablespaces and AUTOALLOCATE for all other types of tablespaces. If you do not specify the EXTENT_MANAGEMENT clause, then Oracle Database interprets the MINIMUM EXTENT clause and the DEFAULT STORAGE clause to determine extent management.

Also with the EXTENT MANAGEMENT clause, you can specify where the metadata for allocated and unallocated extents is to be stored - either in the data dictionary (DICTIONARY) or in the tablespace itself (LOCAL). Tablespaces that record extent allocation in the dictionary, are called dictionary managed tablespaces. Tablespaces that record extent allocation in the tablespace header, are called locally managed tablespaces

Logging

You can include the LOGGING clause to specify the default logging attributes of all tables, indexes, materialized views, materialized view logs, and partitions within a tablespace. The logging attribute controls whether certain DML operations are logged in the redo log file (LOGGING) or not (NOLOGGING). The default is LOGGING. This clause is not valid for a temporary or undo tablespace.

If logging is not enabled, any direct loads using SQL*Loader and direct load INSERT operations are not written to the redo log, and the objects are thus unrecoverable in the event of data loss. When an object is created without logging enabled, you must back up those objects if you want them to be recoverable. Choosing not to enable logging can have a significant impact on the ability to recover objects in the future. Use with caution.

You can use the `FORCE LOGGING` clause to put the tablespace into `FORCE LOGGING` mode. Oracle Database will log all changes to all objects in the tablespace except changes to temporary segments, overriding any `NOLOGGING` setting for individual objects. The database must be open and in `READ WRITE` mode.

Segment Space Management

You can include the `SEGMENT MANAGEMENT` clause to specify whether Oracle Database should track the used and free space in the segments in the tablespace using free lists or bitmaps (`AUTO`) or not (`MANUAL`).

- `AUTO`: Oracle Database server will use bitmaps to manage the free space in segments. The bitmap describes the status of each data block in a segment with respect to the amount of space in the block that is available for inserting rows. As more or less space becomes available in a data block, the new state is reflected in the bitmap. With bitmaps, the Oracle database manages free space more automatically. As a result, this form of space management is called Automatic Segment Space Management (ASSM).
- `MANUAL`: You want to use free lists for managing free space in segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space in segments is called manual segment space management because of the need to specify and tune the `PCTUSED`, `FREELISTS`, and `FREELIST GROUPS` storage parameters for schema objects created in the tablespace. This is supported for backward compatibility; it is recommended that you use ASSM.

The `SEGMENT MANAGEMENT` clause applies to permanent, locally managed tablespaces only, and is not valid for temporary tablespaces.

Data Segment Compression

When you create a tablespace, you can specify that all tables and indexes, or their partitions, created in a tablespace are compressed by default.

Data segment compression is disabled by default. Enabling data segment compression can save disk space usage, reduce memory use in the buffer cache, and speed up query execution during reads. There is, however, a cost in CPU overhead for data loading and DML. It is especially useful in online analytical processing (OLAP) systems, where there are lengthy read-only operations, but can also be used in online transaction processing (OLTP) systems.

View:

Click [CREATE TABLESPACE](#), scroll down, and review the following examples in *Oracle Database SQL Language Reference*:

- Creating Basic Tablespaces
- Enabling `AUTOEXTEND` for a Tablespace
- Creating a Locally-Managed Tablespace
- Creating a Temporary Tablespace
- Creating an Undo Tablespace

Click the following link to *Oracle Database Administrator's Guide* to view an example of creating tablespaces with default compression attributes.

- [Tablespaces with Default Compression Attributes](#)

Altering and Dropping Tablespaces

- When you create a tablespace, it is initially a read/write tablespace.
- Use the ALTER TABLESPACE statement to take a tablespace offline or online, add data files or temp files to it, or make it a read-only tablespace.
- A tablespace can be in one of three different statuses or states:
 - Read Write
 - Read Only
 - Offline with one of the following options:
 - Normal
 - Temporary
 - Immediate
- Add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file.
- Use the DROP TABLESPACE statement to drop a tablespace and its contents from the database if you longer need its content.

ORACLE

- **Immediate:** A tablespace can be taken offline immediately without Oracle Database taking a checkpoint on any of the data files. When you specify Immediate, media recovery for the tablespace is required before the tablespace can be brought online. You cannot take a tablespace offline immediately if the database is running in NOARCHIVELOG mode.

Note: System tablespaces may not be taken offline.

Changing the Size

You can add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file. You cannot add additional data files to bigfile tablespaces. You can make a tablespace either larger or smaller. However, you cannot make a data file smaller than the used space in the file. If you try to do so, you'll get an error.

Dropping Tablespaces

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. You must have the `DROP TABLESPACE` system privilege to drop a tablespace.

When you drop a tablespace, the file pointers in the control file of the associated database are removed. If you are using Oracle Managed Files (OMF), the underlying operating system files are also removed. Otherwise, without OMF, you can optionally direct the Oracle server to delete the operating system files (data files) that constitute the dropped tablespace. If you do not direct the Oracle server to delete the data files at the same time that it deletes the tablespace, you must later use the appropriate commands of your operating system if you want them to be deleted.

You cannot drop a tablespace that contains active segments. For example, if a table in the tablespace is currently being used, or if the tablespace contains undo data that is needed to roll back uncommitted transactions, you cannot drop the tablespace. It is best to take the tablespace offline before dropping it.

Viewing Tablespace Information

Tablespace and data file information can be obtained by querying the following views in SQL*Plus:

- Tablespace information:
 - DBA_TABLESPACES
 - V\$TABLESPACE
- Data file information:
 - DBA_DATA_FILES
 - V\$DATAFILE
- Temp file information:
 - DBA_TEMP_FILES
 - V\$TEMPFILE
- Tables in a tablespace:
 - ALL_TABLES

ORACLE

Implementing Oracle Managed Files

- Specify file operations in terms of database objects rather than file names.

Parameter	Description
DB_CREATE_FILE_DEST	Defines the location of the default file system directory for data files and temporary files
DB_CREATE_ONLINE_LOG_DEST_n	Defines the location for redo log files and control file creation
DB_RECOVERY_FILE_DEST	Gives the default location for the fast recovery area

- Example:

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='/u01/app/oracle/oradata';
SQL> CREATE TABLESPACE tbs_1;
```



About Oracle Managed Files

Oracle Managed Files (OMF) eliminates the need for you to directly manage the operating system files in an Oracle database. You specify operations in terms of database objects rather than file names. The database internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Redo log files
- Control files
- Archived logs
- Block change tracking files
- Flashback logs
- RMAN backups

A database can have a mixture of Oracle-managed and Oracle-unmanaged files. The file system directory specified by either of these parameters must already exist; the database does not create it. The directory must also have permissions for the database to create the files in it.

The table in the slide above lists three initialization parameters that are used with OMF. The example in the slide shows that after the `DB_CREATE_FILE_DEST` initialization parameter is set, you can omit the `DATAFILE` clause from the `CREATE TABLESPACE` statement. The data file is created in the location specified by `DB_CREATE_FILE_DEST`. When you create a tablespace as shown below, default values are assigned to all parameters.

Naming Formats

Oracle-managed files have a specific naming format. For example, on Linux- and UNIX-based systems the following format is used:

```
<destination_prefix>/o1_mf_%t_%u_.dbf
```

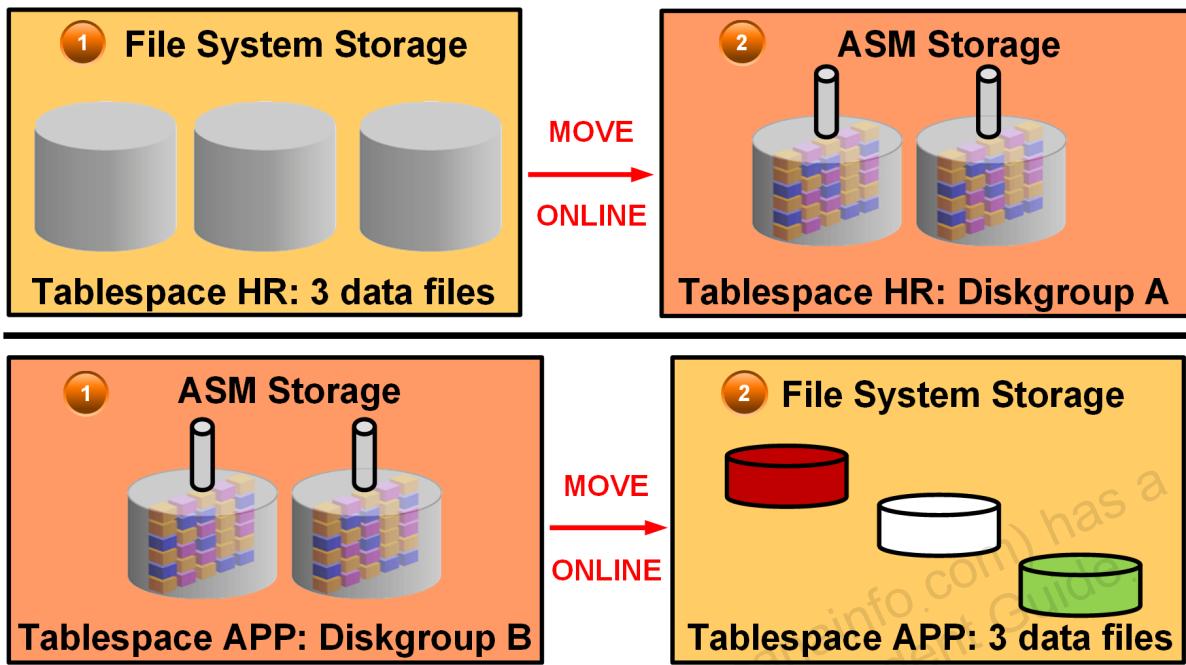
Do not rename an Oracle-managed file. The database identifies an Oracle-managed file based on its name. If you rename the file, the database is no longer able to recognize it as an Oracle-managed file and will not manage the file accordingly.

File Size

By default, Oracle-managed data files, including those for the SYSTEM and SYSAUX tablespaces, are 100 MB and auto-extensible.

Note: By default, Automatic Storage Management (ASM) uses OMF files, but if you specify an alias name for an ASM data file at tablespace creation time or when adding an ASM data file to an existing tablespace, then that file will not be OMF.

Moving or Renaming Online Data Files



ORACLE®

You can rename and move an online data file from one kind of storage system to another while the database is open and accessing the file. The first example in the diagram above illustrates moving the HR tablespace (three data files) from file system storage to ASM storage (diskgroup A). The second example illustrates moving the APP tablespace from ASM storage (diskgroup B) to file system storage (3 data files).

Queries and DML and DDL operations can be performed while the data file is being moved, for example:

- SELECT statements against tables and partitions
- Creation of tables and indexes
- Rebuilding of indexes

Other notes:

- If objects are compressed while the data file is moved, the compression remains the same.
- You do not have to shut down the database or take the data file offline while you move a data file to another location, disk, or storage system.
- You can omit the TO clause only when an Oracle-managed file is used. In this case, the DB_CREATE_FILE_DEST initialization parameter should be set to indicate the new location.
- If the REUSE option is specified, the existing file is overwritten.
- If the KEEP clause is specified, the old file will be kept after the move operation. The KEEP clause is not allowed if the source file is an Oracle-managed file.

- Use the v\$session_longops view to display ongoing online move operations. Each ongoing operation has one row. The state transition of a successful online move operation is usually NORMAL to COPYING to SUCCESS and finally to NORMAL.

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

Examples of Moving and Renaming Online Data Files

- Relocating an online data file:

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf' TO  
' /disk2/myexample01.dbf' ;
```

- Copying a data file from a file system to Automatic Storage Management (ASM):

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf' TO '+DiskGroup2'  
KEEP ;
```

- Renaming an online data file:

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf' TO  
' /disk1/myexample02.dbf' ;
```

Summary for Lesson 6

Unauthorized reproduction or distribution prohibited
Copyright © 2017, Oracle and/or its affiliates.

In this lesson, you should have learned to:

- Explain how table data is stored in the database
- Create tablespaces in SQL*Plus
- Alter or drop tablespaces in SQL*Plus
- View tablespace information in SQL*Plus
- Implement Oracle Managed Files (OMF)
- Move or rename online data files in SQL*Plus



ORACLE

Samad Ul Haq (samad.ulhaq@arcanainformatics.com)
non-transferable license to use this material

Practice 6 Overview

- 6-1: Viewing Tablespace Information with SQL*Plus and EM Express
- 6-2: Creating a Tablespace with SQL Developer and SQL*Plus

Practice 6-1 Viewing Tablespace Information

Overview

In this practice, you use SQL*Plus to query various views to learn about tablespace content in PDB1. You also view tablespace information with Enterprise Manager Database Express (EM Express).

Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

Viewing Tablespace Information with SQL*Plus

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

```
$ . oraenv
ORACLE_SID = [oracle] ? ORCL

The Oracle base has been set to /u01/app/oracle
$
```

2. Start SQL*Plus and connect to PDB1 as the `PDBADMIN` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus PDBADMIN/<password>@PDB1

SQL*Plus: Release 12.2.0.1.0 Production on Thu Jul 21 19:40:55 2016
Copyright (c) 1982, 2016, Oracle. All rights reserved.

Connected to:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
SQL>
```

3. List the columns in the `DBA_TABLESPACES` view by using the `DESCRIBE` command. This view describes all tablespace information in the database. The results show that this view contains many columns.

```
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
MAX_EXTENTS		NUMBER
MAX_SIZE		NUMBER
PCT_INCREASE		NUMBER
MIN_EXTLEN		NUMBER
STATUS		VARCHAR2(9)
CONTENTS		VARCHAR2(21)
LOGGING		VARCHAR2(9)
FORCE_LOGGING		VARCHAR2(3)
EXTENT_MANAGEMENT		VARCHAR2(10)
ALLOCATION_TYPE		VARCHAR2(9)
PLUGGED_IN		VARCHAR2(3)
SEGMENT_SPACE_MANAGEMENT		VARCHAR2(6)
DEF_TAB_COMPRESSION		VARCHAR2(8)
RETENTION		VARCHAR2(11)
BIGFILE		VARCHAR2(3)
PREDICATE_EVALUATION		VARCHAR2(7)
ENCRYPTED		VARCHAR2(3)
COMPRESS_FOR		VARCHAR2(30)
DEF_INMEMORY		VARCHAR2(8)
DEF_INMEMORY_PRIORITY		VARCHAR2(8)
DEF_INMEMORY_DISTRIBUTE		VARCHAR2(15)
DEF_INMEMORY_COMPRESSION		VARCHAR2(17)
DEF_INMEMORY_DUPLICATE		VARCHAR2(13)
SHARED		VARCHAR2(12)
DEF_INDEX_COMPRESSION		VARCHAR2(8)
INDEX_COMPRESS_FOR		VARCHAR2(13)
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>		

4. List the tablespaces in PDB1.

```
SQL> SELECT DISTINCT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
-----
SYSAUX
UNDOTBS1
TEMP
USERS
SYSTEM
SQL>
```

5. Find out which tablespace contains the HR schema by querying the ALL_TABLES view. The result shows that the SYSAUX tablespace contains the HR schema.

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

TABLESPACE_NAME
-----
SYSAUX
SQL>
```

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

- STATUS is ONLINE, so the tablespace is available to users. CONTENTS equals the PERMANENT tablespace type.
- LOGGING equals LOGGING, which means certain DML operations are logged in the redo log file.
- PLUGGED_IN equals NO, which tells you that the tablespace is not plugged in.
- BIGFILE equals NO, which indicates that the tablespace is a smallfile tablespace.
- EXTENT_MANAGEMENT equals LOCAL, which means the tablespace is locally managed (not dictionary managed).
- ALLOCATION_TYPE equals SYSTEM, which tells you that the extents of the tablespace are managed by the system, and you cannot specify an extent size.

Note: The results below are formatted for easier viewing.

```
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>
```

7. 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

8. Query the V\$TABLESPACE view for the SYSAUX tablespace.

- INCLUDED_IN_DATABASE_BACKUP equals YES, which means the tablespace is included in full database backups using the BACKUP DATABASE RMAN command.
- BIGFILE equals NO, which means the tablespace is a smallfile tablespace.
- FLASHBACK_ON equals YES, which indicates that the tablespace participates in FLASHBACK DATABASE operations.
- ENCRYPT_IN_BACKUP equals null, which means that encryption is neither explicitly turned on or off at the tablespace level.
- CON_ID refers to the container to which the data pertains. In this case, PDB1 is container ID 3.

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

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

9. List all the tables in the SYSAUX tablespace. There are hundreds of tables so restrict the query to tables owned by the HR account.

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

TABLE_NAME
REGIONS
LOCATIONS
DEPARTMENTS
JOBS
EMPLOYEES
JOB_HISTORY

6 rows selected.

10. List all the indexes in the SYSAUX tablespace. Again, there are hundreds so restrict the query to indexes owned

by the HR account. The SET PAGES command specifies 100 rows to print per page.

```
SQL> SET PAGES 100
SQL> SELECT index_name FROM all_indexes WHERE tablespace_name='SYSAUX' AND
owner='HR';

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

19 rows selected.
SQL>
```

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

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

SQL>
```

12. List data file information for the SYSAUX tablespace by querying various columns in the DBA_DATA_FILES view. The results show the following:

- AUTOEXTENSIBLE is equal to YES, which tells you that the auto extend feature is enabled for a data file. The tablespace size can increase without you having to take any action.
- BYTES is equal to 377487360 (your value may differ), which is the size of the file in bytes.
- MAXBYTES is equal to 3.4360E+10 (your value may differ), which is the maximum file size allowed.
- USER_BYTES is equal to 376438784, which is the size of the file available for user data. The actual size of the file minus the USER_BYTES value is used to store file-related metadata.

Note: The results below are formatted for easier viewing.

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

FILE_NAME                      AUT      BYTES    MAXBYTES  USER_BYTES
-----  -----
/u01/app/oracle/oradata/ORCL/    YES    377487360  3.4360E+10 376438784
PDB1/sysaux01.dbf
SQL>
```

13. Find out how many segments there are in the SYSAUX tablespace by querying the DBA_SEGMENTS view. The results show that there are 1606 segments. Your number may vary.

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

COUNT(Segment_Name)
-----
1606
SQL>
```

14. Find out which index in the SYSAUX tablespace takes up the most space by querying the DBA_SEGMENTS view. The results indicate that the I_WRI\$\$_OPTSTAT_H_OBJ#_ICOL#_ST index takes up the most space. The results below are formatted for easier viewing. Your bytes value will be different than the one shown below.

```

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      8388608
SQL>

```

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

```

SQL> EXIT
...
$
```

Viewing Tablespace Information with Enterprise Manager Database Express

1. Open a Firefox web browser, and enter the URL <https://localhost:5500/em> to connect the client browser to PDB1 in EM Express.
2. On the Login page for EM Express, enter the user name **PDBADMIN**, enter the password as specified in **Product-Specific Credentials**, enter **PDB1** for the container name, and click **Login**. The Database Home page is displayed.

3. Select **Storage**, and then **Tablespaces**. All the tablespaces are listed with their size, amount of free space, amount used (%), Auto Extend setting, Maximum Size setting, Status, Type, Group Name, Auto Segment Management setting, and Directory. Your values will be different than the ones shown below.

Name	Size	Free Space	Used (%)	Auto...	Ma...	Status	Type	Group...	Auto...	Directory
SYSAUX	360MB	19MB	94.8	✓	Unlimi	Green	File		✓	/u01/app/oracle/ora...
SYSTEM	250MB	6MB	97.8	✓	Unlimi	Green	File			/u01/app/oracle/ora...
TEMP	64MB	63MB	1.6	✓	Unlimi	Green	File			/u01/app/oracle/ora...
UNDOTBS1	100MB	60MB	40.1	✓	Unlimi	Green	File			/u01/app/oracle/ora...
USERS	5MB	2MB	51.2	✓	Unlimi	Green	File		✓	/u01/app/oracle/ora...

4. Question: How much of the SYSAUX tablespace is already used?

Answer: According to the table above, 94.8% of the SYSAUX tablespace has already been used. It has 19MB of free space left.

5. Log out of EM Express, and close the browser window.

Practice 6-2 Creating a Tablespace

Overview

In this practice, you create and populate a tablespace named INVENTORY. You use SQL Developer to generate a SQL script to create the tablespace, but you use SQL*Plus to execute the SQL script.

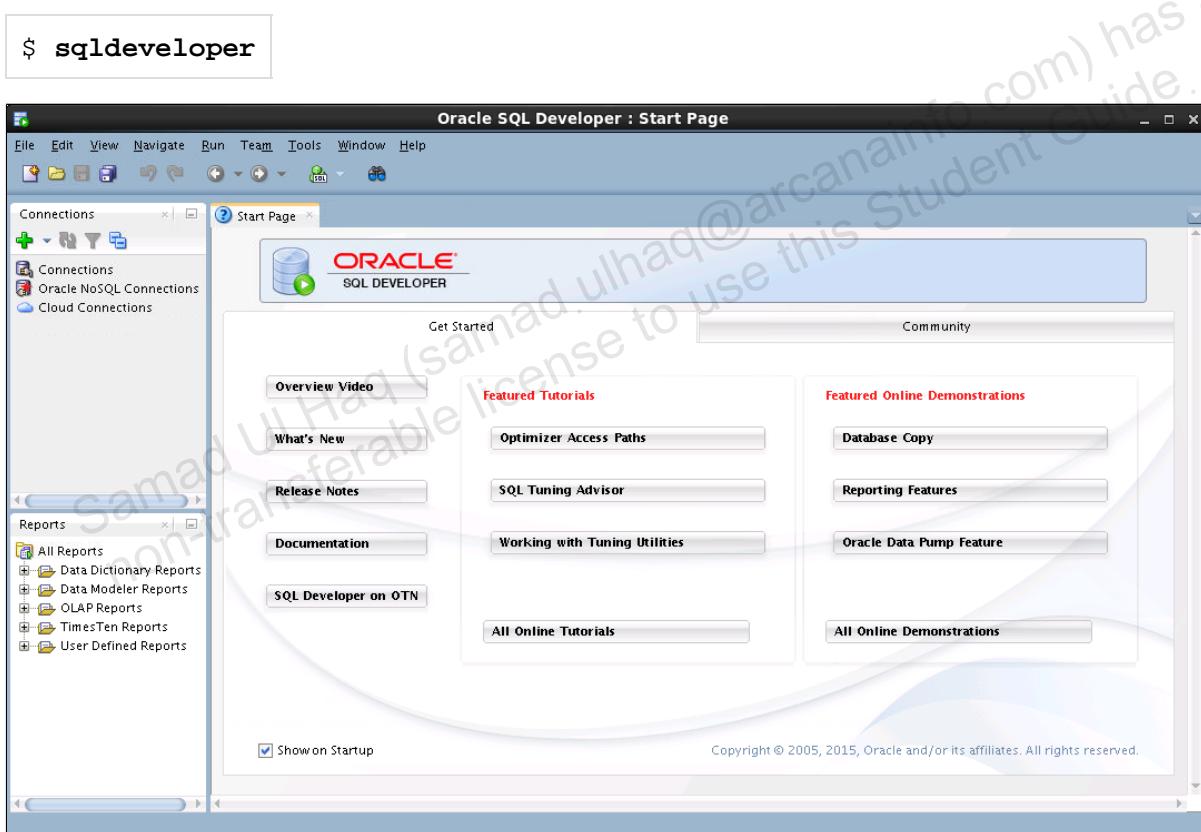
Assumptions

You are logged in to VM1 as the `oracle` user.

Tasks

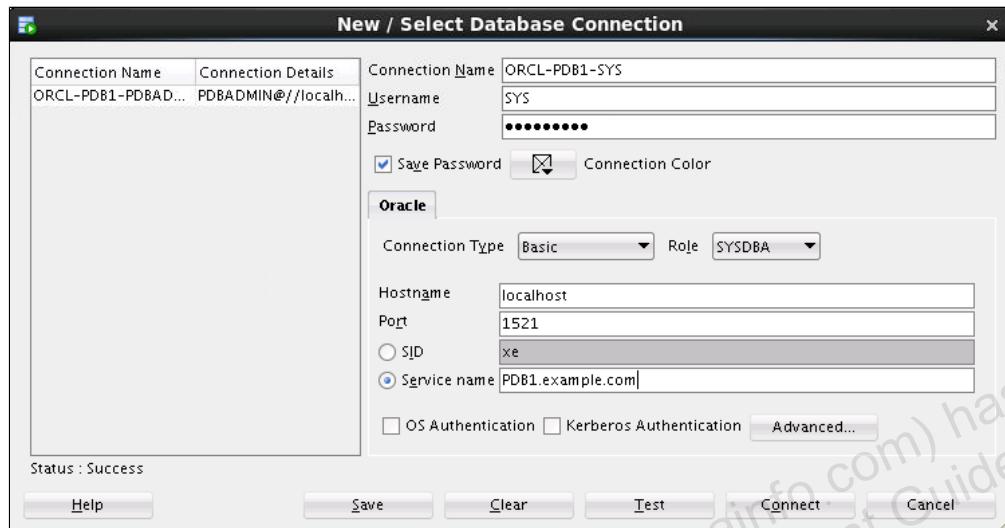
[View Information About PDB1 with SQL Developer](#)

1. Open a terminal window and launch SQL Developer. The Oracle SQL Developer: Start Page tab is displayed. Alternatively, you can double-click the **sqldeveloper** shortcut on your desktop.

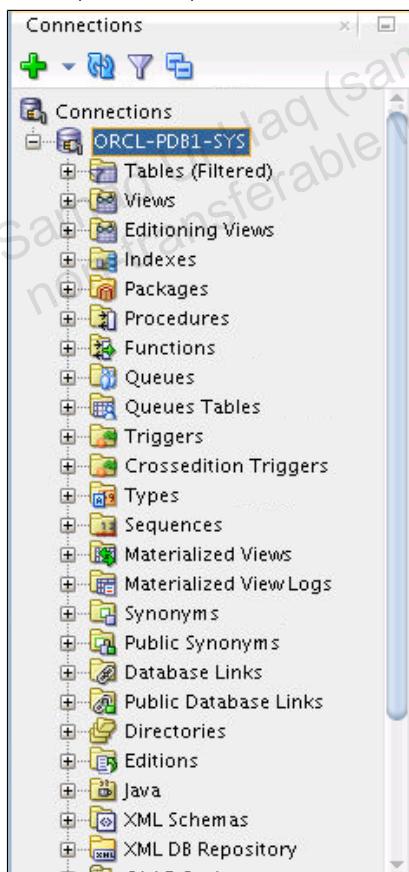


2. Create a connection to PDB1 for the SYS user.
 - a. In the Connections pane, click the **New Connection** (plus sign) button. The New/Select Database Connection dialog box is displayed.
 - b. Configure the following information, and click **Test**. The Status in the bottom left corner should read Success.
 - a. Connection Name: **ORCL-PDB1-SYS** (This example uses the naming convention: *instance name-container name-user name*.)
 - b. Username: **SYS**

- c. Password: See Appendix - Product-Specific Credentials for the password.
- d. Save Password: Select this check box
- e. Connection Type: **Basic**
- f. Role: **SYSDBA**
- g. Hostname: **localhost**
- h. Port: **1521**
- i. Service name: **PDB1.example.com**

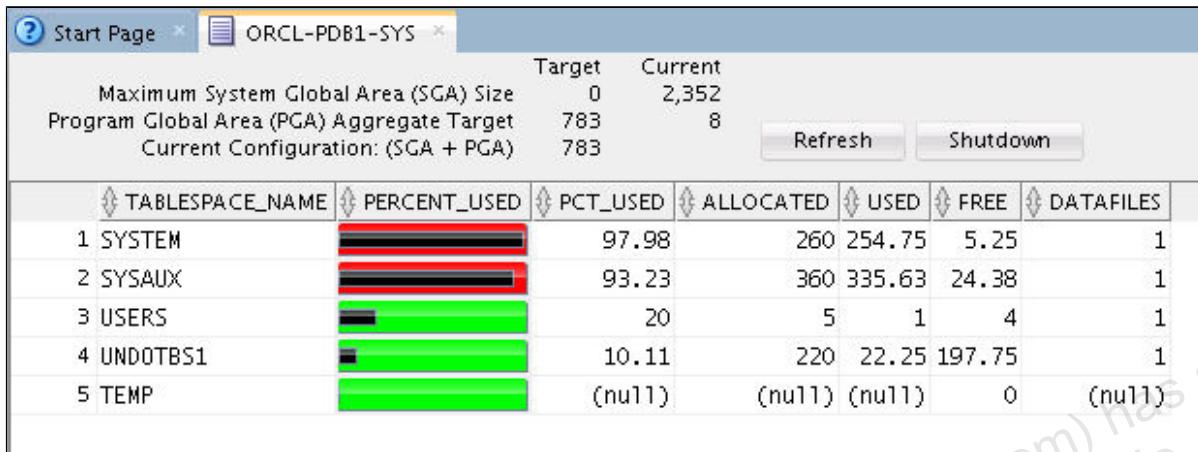


- c. Click **Save**, and then **Connect**.
- d. In the Connections pane, expand **ORCL-PDB1-SYS**. The list is populated with items, such as tables, views, indexes, and so on.



e. Expand a few items in the list, for example, **Tables (Filtered)** and **Views**. Only those objects that are owned by the **SYS** user are listed.

3. In the Connections pane, right-click **ORCL-PDB1-SYS**, and select **Manage Database**. On the right, the ORCL-PDB1-SYS tab shows you information about the SGA, PGA, and tablespaces of PDB1. Note: Tablespaces do not belong to any user, just like rollback segments and roles. For each tablespace, you can view the percentage of storage that is already used, the amount of storage allocated, the amount of storage already used, the amount of storage that is still available, and the number of data files associated with the tablespace.

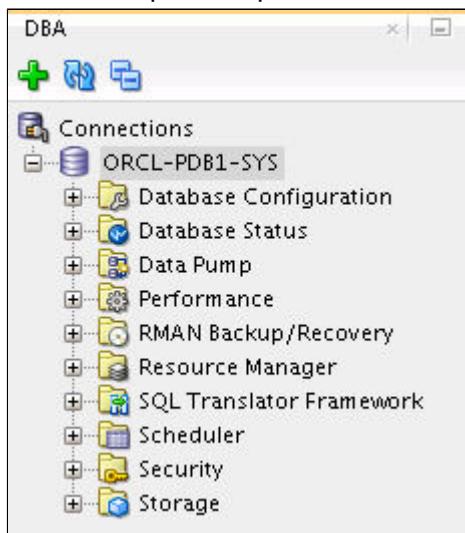


4. Add a connection to the DBA navigator.

- On the main menu, select **View**, and then **DBA** to display the DBA panel.
- In the DBA pane at the bottom left, click the **Add Connection** button (plus sign). In the Select Connection dialog box, leave **ORCL-PDB1-SYS** selected as the connection, and click **OK**.



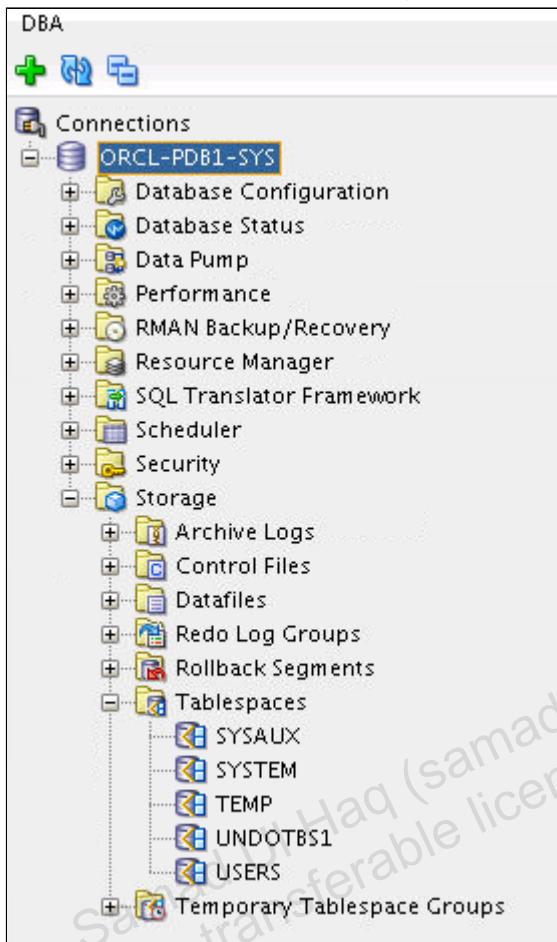
- In the DBA pane, expand **ORCL-PDB1-SYS**, and view the database administration options below.



Generate the SQL Code to Create an INVENTORY Tablespace

In this section, you use SQL Developer to generate the SQL code needed to create a tablespace called `INVENTORY`. Even though you could, you don't create the tablespace in SQL Developer. Instead, you copy the SQL code to a file and execute it later in the practice.

1. In the DBA pane, expand **Storage**, and then **Tablespaces**. The tablespaces of PDB1 are listed.

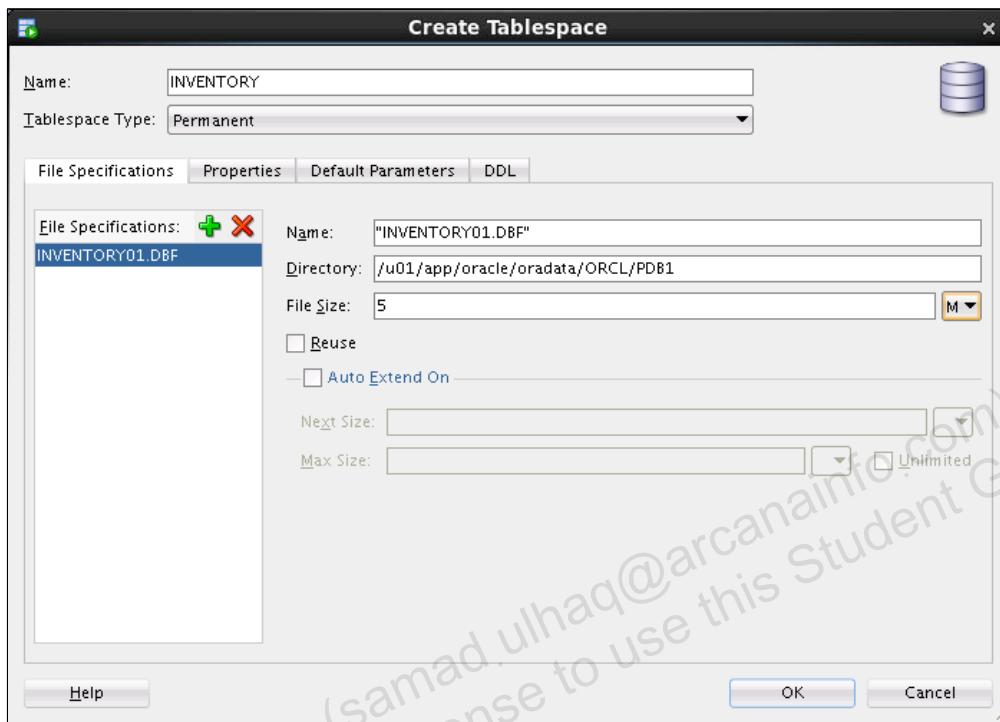


2. In the DBA pane, select **Tablespaces**. A Tablespaces tab is displayed on the right. View the information. Each tablespace is listed with its allocated amount of storage, free storage amount, used storage amount, percentage of free storage, percentage of used storage, and maximum amount of storage.

TABLESPACE_NAME	MEGS_ALLOC	MEGS_FREE	MEGS_USED	PCT_FREE	PCT_USED	MAX
1 USERS	5	4	1	80	20	32768
2 TEMP	64	64	0	100	0	32768
3 UNDOTBS1	100	84	16	84	16	32768
4 SYSTEM	260	7	253	3	97	32768
5 SYSAUX	420	24	396	6	94	32768

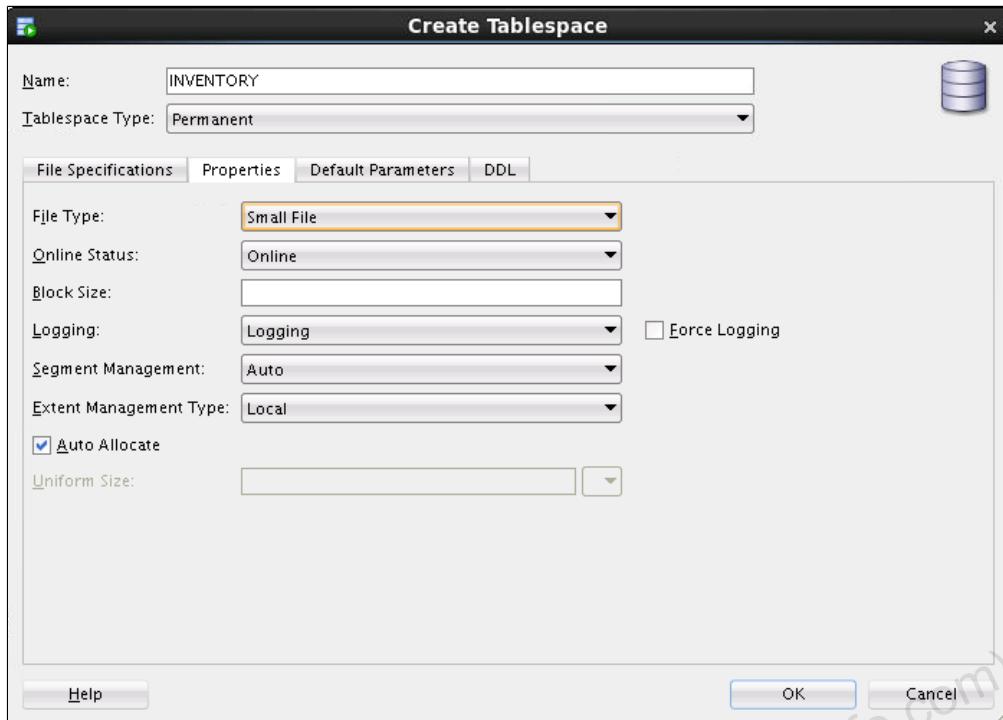
3. To create a new tablespace, on the Tablespaces tab, select **Actions**, and then **Create New**. The Create Tablespace window is displayed.
4. In the Name box, enter `INVENTORY`. This will be the name of your tablespace.

5. In the Tablespace Type drop-down list, leave **Permanent** selected.
6. On the File Specifications tab, configure the following:
 - Name (for data file): **INVENTORY01.DBF** (You use a numbering system in the name because you will later add another data file named **INVENTORY02.dbf** to this tablespace.)
 - Directory: **/u01/app/oracle/oradata/ORCL/PDB1**
 - File Size: **5MB** (Select **M** from the drop-down list.)
 - Reuse: Do not select this check box.
 - Auto Extend On: Do not select this check box.



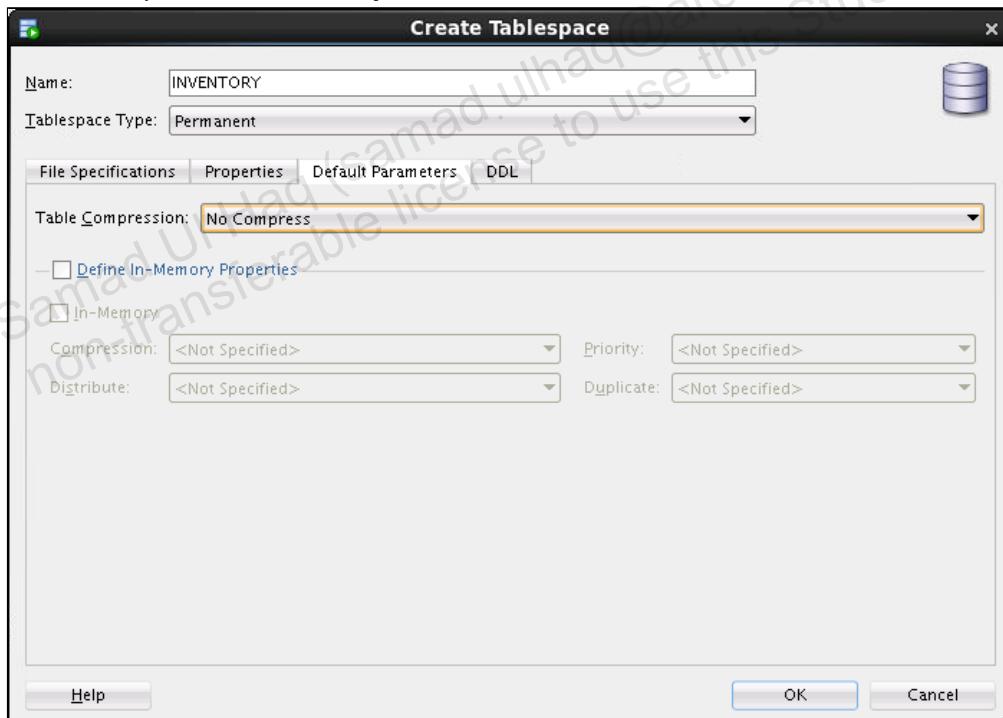
7. Click the **Properties** tab and ensure the following configuration is set:
 - File Type: **Small File**
 - Online Status: **Online**
 - Logging: **Logging** (Leave the Force Logging check box deselected.)
 - Segment Management: **Auto**
 - Extent Management Type: **Local**

- Auto Allocate (extent allocation): Select this check box.

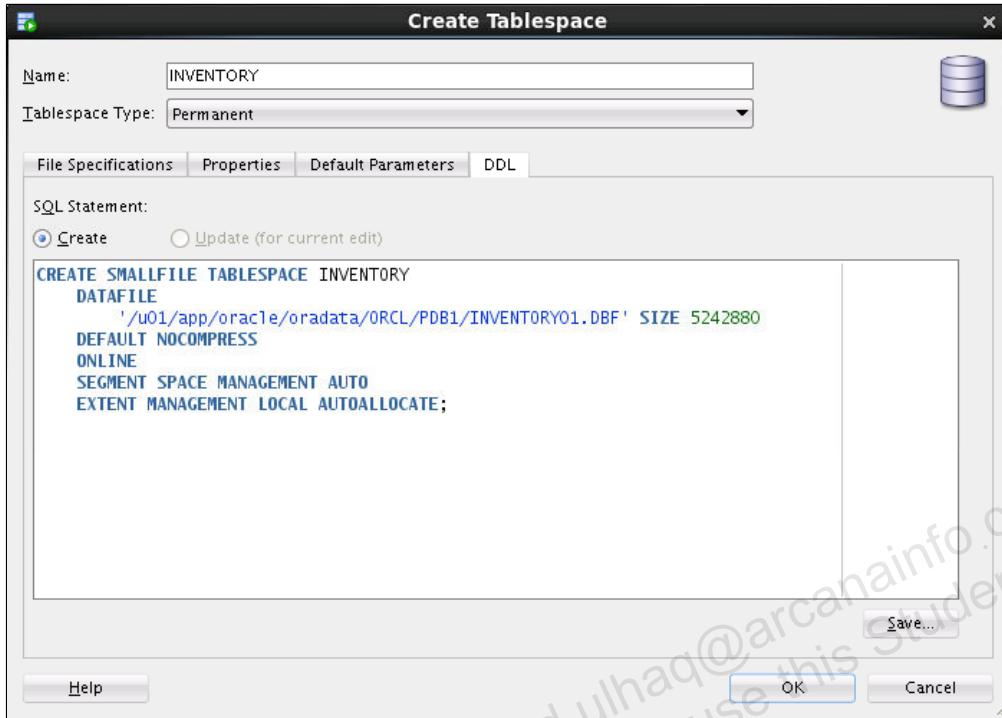


8. Click the **Default Parameters** tab and ensure the following configuration is set:

- Table Compression: **No Compress**



9. Click the **DDL** tab, and then click the **Create** option and review the SQL code. Next, press **Ctrl+c** to copy the SQL to the clipboard. Leave this window open for a moment, but don't click OK. You will eventually cancel out of here.



10. From the main menu on your desktop, select **Applications**, then **Accessories**, and then **gedit Text Editor**. A new document is created in gedit.
11. Press **Ctrl+v** to paste the SQL into the document. Review the SQL code and make sure that it ends with a semicolon (;). The SQL code should read as follows. Make sure you have the correct path for the data file.

```
CREATE SMALLFILE TABLESPACE INVENTORY  
  
DATAFILE '/u01/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF' SIZE 5242880  
DEFAULT NOCOMPRESS  
ONLINE  
SEGMENT SPACE MANAGEMENT AUTO  
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

12. Select **File**, and then **Save As**. In the Name box, enter `CreateINVENTORYTablespace.sql`. Leave the `oracle` folder selected. Click **Save**.
13. Select **File**, and then **Quit** to exit gedit.
14. Now that you have your SQL code safely saved in a file, click **Cancel** in the Create Tablespace window in SQL Developer.
15. Select **File**, and then **Exit** to exit SQL Developer. You are returned to the terminal window in which you will continue to work.

Create the INVENTORY Tablespace and Table X in SQL*Plus

As the PDBADMIN user in SQL*Plus, run the script you just created to create the INVENTORY tablespace. Next, run a pre-created 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 walk through the steps to correct the problem.

1. In the terminal window, source the oraenv script. For the ORACLE_SID value, enter ORCL .

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

2. Start SQL*Plus and connect to PDB1 as the SYS user with the SYSDBA privilege. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYS/<password>@PDB1 AS SYSDBA  
  
...  
SQL>
```

3. Grant the DBA role to the PDBADMIN user. This role enables PDBADMIN to create tablespaces and tables. Note: If you completed [Practice 5-3 Granting the DBA Role to PDBADMIN](#), then you can skip this step.

```
SQL> GRANT DBA TO PDBADMIN;  
  
Grant succeeded.  
SQL>
```

4. Connect to PDB1 as the PDBADMIN user. See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CONNECT PDBADMIN/<password>@PDB1  
  
Connected.  
SQL>
```

5. Run the `CreateINVENTORYTablespace.sql` script, which you created earlier in this practice, to create the INVENTORY tablespace.

```
SQL> @/home/oracle/CreateINVENTORYTablespace.sql  
  
Tablespace created.  
SQL>
```

6. Run the `CreateTable_X.sql` script to create and populate the X table. Notice that near the end, you get an error message: unable to extend table PDBADMIN.X by 128 in tablespace INVENTORY. You get this message because the tablespace in which you are trying to create table X is too small. You will remedy this problem in the next section.

```

SQL> @/home/oracle/labs/CreateTable_X.sql

...
SQL> INSERT INTO x
  2  SELECT * FROM x ;

1024 rows created.

SQL> INSERT INTO x select * from 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.

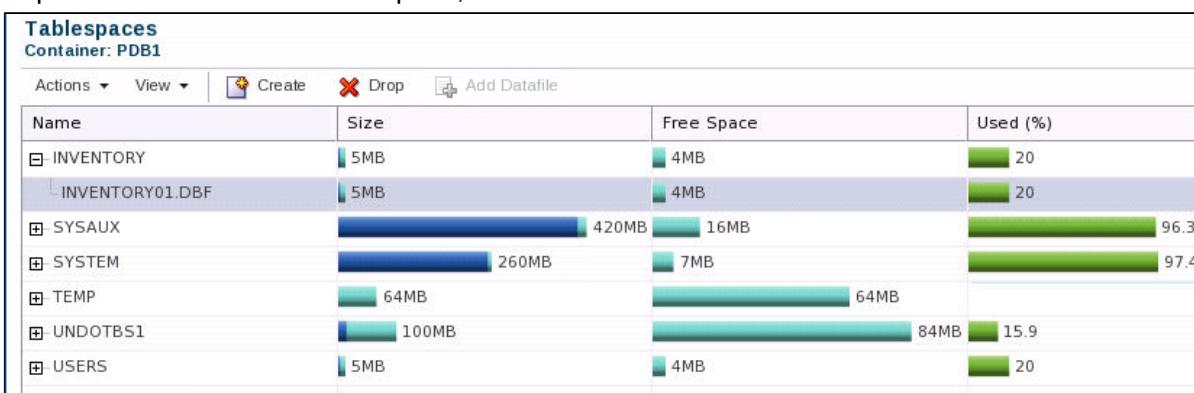
SQL> quit
Disconnected ...
$
```

7. Close the terminal window.

Increase the Size of the INVENTORY01.DBF Data File

Fix the problem that you encountered in the previous section by increasing the size of the INVENTORY01.DBF data file. Use EM Express because it provides an easy-to-use interface when working with tablespaces.

1. Connect the client browser to PDB1 in EM Express.
 - a. Open Firefox and enter the URL <https://localhost:5500/em>.
 - b. On the Login page for EM Express, enter the user name **PDBADMIN** and the password as specified in [Appendix - Product-Specific Credentials](#). Enter **PDB1** for the container name. Click **Login**.
2. Select **Storage**, and then **Tablespaces**.
3. Expand the **INVENTORY** tablespace, and select the **INVENTORY01.DBF** data file.



4. Select **Actions**, and then **Resize**. The "Resize Datafile INVENTORY01.DBF" window is displayed.

5. In the Size box, enter **40M**. Don't click **OK** just yet.



6. Click the **Show SQL** button to view the SQL code that performs the resize action. The SQL generated is as follows.

```
ALTER DATABASE DATAFILE  
'/u01/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF' RESIZE 40M;
```

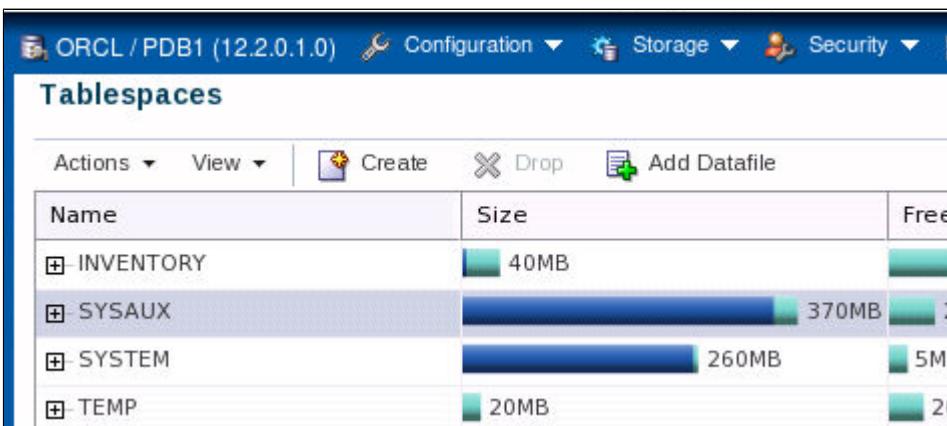
7. In the Confirmation dialog box, click **OK**.

8. In the "Resize Datafile INVENTORY01.DBF" dialog box, click **OK**. The SQL code is executed and the data file is successfully resized.

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

Samad Ul Haq (samad.ulhaq@arcanainfo.com) has a
non-transferable license to use this Student Guide.

10. Verify that the change is reflected in the EM Express interface. The size for the INVENTORY tablespace should now be set to 40MB.

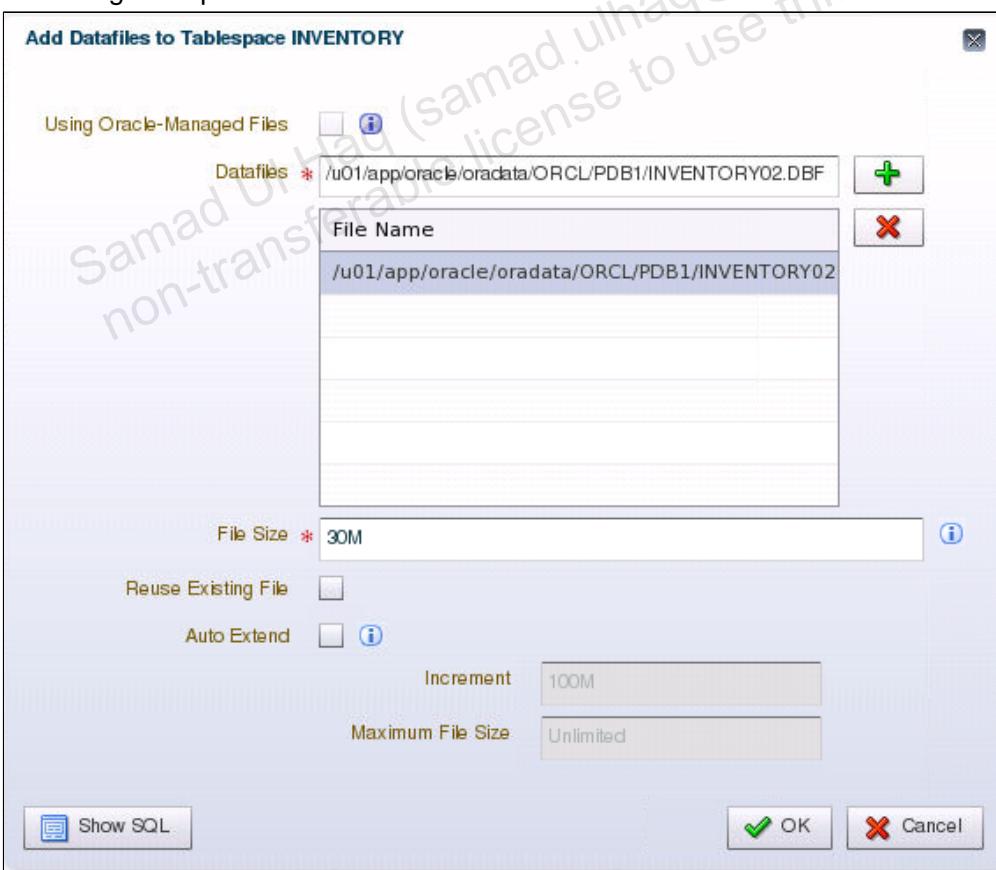


Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Add a Data File to the INVENTORY Tablespace

Continue to work in EM Express.

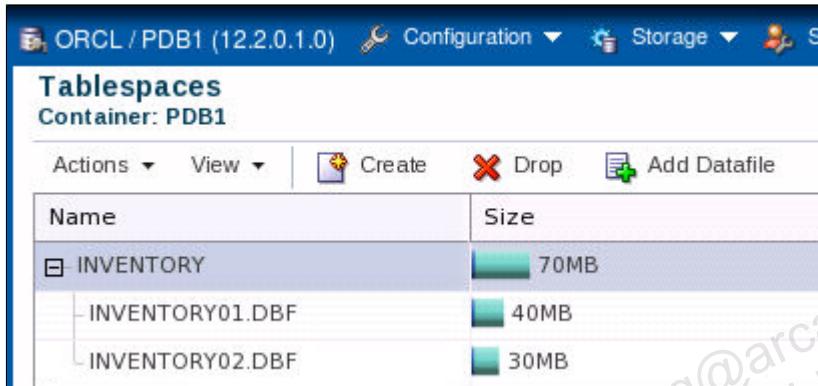
1. Select the INVENTORY tablespace.
2. Select **Actions**, and then **Add Datafile**. The "Add Datafiles to Tablespace INVENTORY" dialog box is displayed.
3. In the Datafiles box, enter `/u01/app/oracle/oradata/ORCL/PDB1/INVENTORY02.DBF`, and click the plus sign to add the file to the list. In the File Size box, enter **30M**. Deselect the **Auto Extend** check box. Leave the dialog box open.



4. Click **Show SQL** and view the SQL code being generated. A Confirmation dialog box displays the following code:

```
ALTER TABLESPACE "INVENTORY" ADD DATAFILE  
' /u01/app/oracle/oradata/ORCL/PDB1/INVENTORY02.DBF' SIZE 30M  
  
AUTOEXTEND OFF;
```

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



9. Click **Log Out** and close the browser window.

Re-Create Table X and Populate It

As the PDBADMIN user, run the script named `CreateTableX.sql` again in SQL*Plus to create and populate the table called X in the INVENTORY tablespace. The script drops the X table and re-executes the original script that previously returned the space error. This time, however, you shouldn't receive an error because you increased the size of the tablespace.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`.

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

2. Start SQL*Plus and connect to PDB1 as the PDBADMIN user. See [Appendix - Product-Specific Credentials](#) for the password.

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

3. Run the `CreateTable_X.sql` script, located in `/home/oracle/labs`. The script runs without any errors.

```
SQL> @/home/oracle/labs/CreateTable_X.sql

...
SQL> INSERT INTO x
  2  SELECT * FROM x ;

1024 rows created.

SQL> INSERT INTO x
  2  SELECT * FROM x ;

2048 rows created.

SQL> COMMIT;

Commit complete.

SQL> quit
$
```

4. Start SQL*Plus again and connect to `PDB1` as the `PDBADMIN` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus PDBADMIN/<password>@PDB1

...
SQL>
```

5. Verify that table `x` was created in the `INVENTORY` tablespace.

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

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

Drop the INVENTORY Tablespace

1. Drop the `INVENTORY` tablespace.

```
SQL> DROP TABLESPACE inventory INCLUDING CONTENTS AND DATAFILES;

Tablespace dropped.
SQL>
```

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

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