



Integrated Cloud Applications & Platform Services



Oracle Database 12c R2: Administration Workshop

Student Guide – Volume II

D78846GC30

Edition 3.0 | March 2017 | D99462

Learn more from Oracle University at education.oracle.com

ORACLE®

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 Bopaiah

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

7

Managing Storage Space



ORACLE®

Objectives for Lesson 7

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

After completing this lesson, you should be able to:

- Describe how the Oracle Database server automatically manages space
- Save space by using compression
- Proactively monitor and manage tablespace space usage
- Describe segment creation in the Oracle database
- Control deferred segment creation
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation



ORACLE®

Space Management Features

- Oracle Managed Files (OMF)
- Free-space management with bitmaps (“locally managed”) and automatic data file extension
- Proactive space management (default thresholds and server-generated alerts)
- Space reclamation (shrinking segments, online table redefinition)
- Capacity planning (growth reports)



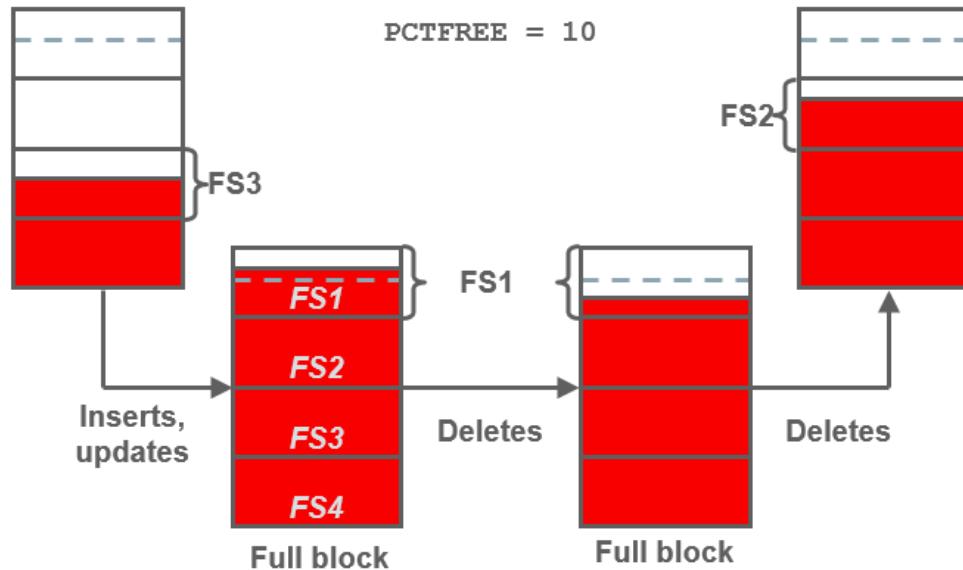
Space is automatically managed by the Oracle Database server. It generates alerts about potential problems and recommends possible solutions.

With Oracle Managed Files (OMF), you can specify operations in terms of database objects rather than file names. The Oracle Database server can manage free space within a tablespace with bitmaps. This is known as a “locally managed” tablespace. In addition, free space within segments located in locally managed tablespaces can be managed using bitmaps. This is known as Automatic Segment Space Management. The bitmapped implementation eliminates much space-related tuning of tables, while providing improved performance during peak loads. Additionally, the Oracle Database server provides automatic extension of data files, so the files can grow automatically based on the amount of data in the files.

When you create a database, proactive space monitoring is enabled by default. (This causes no performance impact.) The Oracle Database server monitors space utilization during normal space allocation and deallocation operations and alerts you if the free space availability falls below the predefined thresholds (which you can override). Advisors and wizards assist you with space reclamation.

For capacity planning, the Oracle Database server provides space estimates based on table structure and number of rows and a growth trend report based on historical space utilization stored in the Automatic Workload Repository (AWR).

Block Space Management



ORACLE®

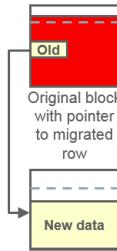
Space management involves the management of free space at the block level. With *Automatic Segment Space Management*, each block is divided into four sections, named FS1 (between 0 and 25% of free space), FS2 (25% to 50% free), FS3 (50% to 75% free), and FS4 (75% to 100% free).

Depending on the level of free space in the block, its status is automatically updated. That way, depending on the length of an inserted row, you can tell whether a particular block can be used to satisfy an insert operation. Note that a “full” status means that a block is no longer available for inserts.

In the example above, the block on the left is an FS3 block because it has between 50% and 75% free space. After some insert and update statements, `PCTFREE` is reached (the dashed line) and it is no longer possible to insert new rows in that block. The block is now considered as a “full” or FS1 block. The block is considered for insertion again, as soon as its free space level drops below the next section. In the preceding case, it gets status FS2 as soon as the free space is more than 25%.

Note: Large object (LOB) data types (`BLOB`, `CLOB`, `NCLOB`, and `BFILE`) do not use the `PCTFREE` storage parameter. Uncompressed and OLTP-compressed blocks have a default `PCTFREE` value of 10; basic compressed blocks have a default `PCTFREE` value of 0.

Row Chaining and Migration



- On update: Row length increases, exceeding the available free space in the block.
- Data needs to be stored in a new block.
- Original physical identifier of row (`ROWID`) is preserved.
- The Oracle Database server needs to read two blocks to retrieve data.
- The Segment Advisor finds segments containing the migrated rows.
- There is automatic coalescing of fragmented free space inside the block.

ORACLE®

In two circumstances, the data for a row in a table may be too large to fit into a single data block. In the first case, the row is too large to fit into one data block when it is first inserted. In this case, the Oracle Database server stores the data for the row in a chain of data blocks (one or more) reserved for that segment. Row chaining most often occurs with large rows, such as rows that contain a column of data type `LONG` or `LONG RAW`. Row chaining in these cases is unavoidable.

However, in the second case, a row that originally fit into one data block is updated, so that the overall row length increases, and the block's free space is already completely filled. In this case, the Oracle Database server migrates the data for the entire row to a new data block, assuming that the entire row can fit in a new block. The database preserves the original row piece of a migrated row to point to the new block containing the migrated row. The `ROWID` of a migrated row does not change.

When a row is chained or migrated, input/output (I/O) performance associated with this row decreases because the Oracle Database server must scan more than one data block to retrieve the information for the row.

The Segment Advisor finds the segments containing migrated rows that result from an `UPDATE`.

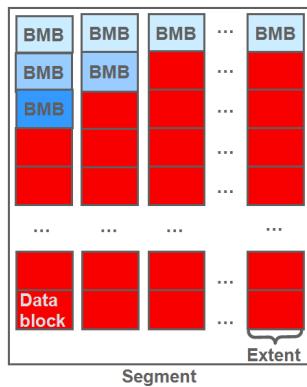
The Oracle Database server automatically and transparently coalesces the free space of a data block when:

- An `INSERT` or `UPDATE` statement attempts to use a block with sufficient free space for a new row piece
- The free space is fragmented, so that the row piece cannot be inserted in a contiguous section of the block

After coalescing, the amount of free space is identical to the amount before the operation, but the space is now contiguous.

Free Space Management Within Segments

Tracked by bitmaps in segments



Benefits:

- More flexible space utilization
- Runtime adjustment
- Multiple process search of BMBs

ORACLE®

Free space can be managed automatically inside database segments. The in-segment free or used space is tracked with bitmaps. To take advantage of this feature, specify Automatic Segment Space Management when you create a locally managed tablespace. Your specification then applies to all segments subsequently created in this tablespace.

Automatic space management segments have a set of bitmap blocks (BMBs) describing the space utilization of the data blocks in that segment. BMBs are organized in a tree hierarchy. The root level of the hierarchy, which contains references to all intermediate BMBs, is stored in the segment header. The leaves of this hierarchy represent the space information for a set of contiguous data blocks that belong to the segment. The maximum number of levels inside this hierarchy is three.

Benefits of using automatic space management include:

- Better space utilization, especially for the objects with highly varying row sizes
- Better runtime adjustment to variations in concurrent access
- Better multi-instance behavior in terms of performance or space utilization

Types of Segments

- A segment is a set of extents allocated for a certain logical structure.
- The different types of segments include:
 - Table and cluster
 - Index
 - Undo
 - Temporary
- Segments are dynamically allocated by the Oracle Database server.



Table and cluster segments: Each nonclustered table has a data segment. All table data is stored in the extents of the table segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.

Index segment: Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.

Undo segment: Oracle Database maintains information to reverse changes made to the database. This information consists of records of the actions of transactions, collectively known as undo. Undo is stored in undo segments in an undo tablespace.

Temporary segment: A temporary segment is created by the Oracle Database server when a SQL statement needs a temporary database area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.

The Oracle Database server dynamically allocates space when the existing extents of a segment become full. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

Allocating Extents

- Searching the data file's bitmap for the required number of adjacent free blocks
- Sizing extents with storage clauses:
 - UNIFORM
 - AUTOALLOCATE
- Viewing extent map
- Obtaining deallocation advice



THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

With locally managed tablespaces, the Oracle Database server looks for free space to allocate to a new extent by first determining a candidate data file in the tablespace and then searching the data file's bitmap for the required number of adjacent free blocks. If that data file does not have enough adjacent free space, then the Oracle Database server looks in another data file.

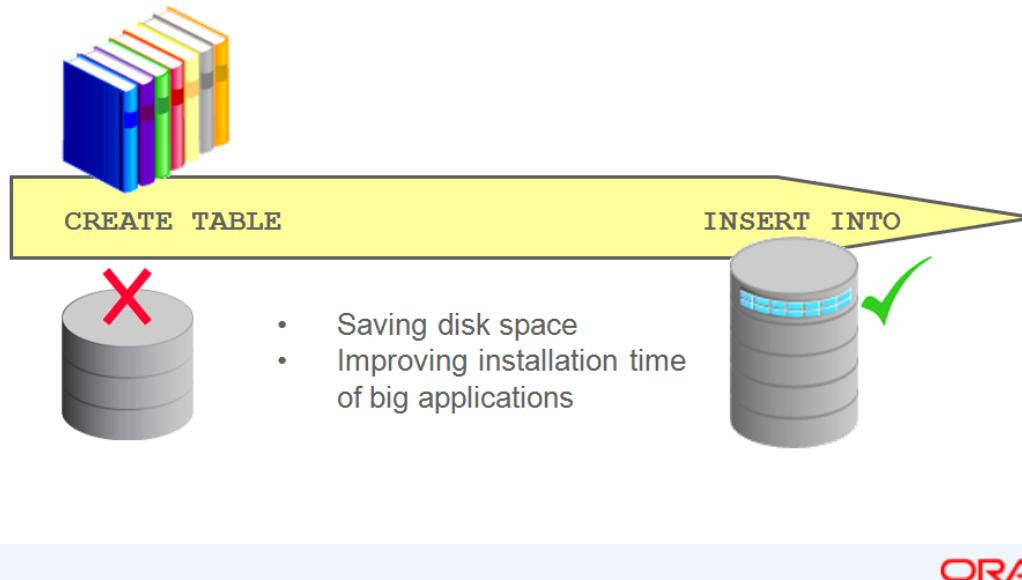
Two clauses affect the sizing of extents:

- With the `UNIFORM` clause, the database creates all extents of a uniform size that you specified (or a default size) for any objects created in the tablespace.
- With the `AUTOALLOCATE` clause, the database determines the extent-sizing policy for the tablespace.

The Oracle Database server provides a Segment Advisor that helps you determine whether an object has space available for reclamation on the basis of the level of space fragmentation within the object.

Deferred Segment Creation

- DEFERRED_SEGMENT_CREATION = TRUE is the default.
- Deferred segment is the default for tables, indexes, and partitions.
- Segment creation takes place as follows:
 1. Table creation > Data dictionary operation
 2. DML > Segment creation



ORACLE®

When you create a nonpartitioned heap table, table segment creation is deferred to the first row insert. This functionality is enabled by default with the DEFERRED_SEGMENT_CREATION initialization parameter set to TRUE.

Advantages of this space allocation method:

- A significant amount of disk space can be saved for applications that create hundreds or thousands of tables upon installation, many of which might never be populated.
- The application installation time is reduced.

When you insert the first row into the table, the segments are created for the base table, its LOB columns, and its indexes. During segment creation, cursors on the table are invalidated. These operations have a small additional impact on performance.

With this allocation method, it is essential that you do proper capacity planning so that the database has enough disk space to handle segment creation when tables are populated.

Controlling Deferred Segment Creation

- With the DEFERRED_SEGMENT_CREATION parameter:
 - Initialization parameter file
 - ALTER SESSION command
 - ALTER SYSTEM command
- With the SEGMENT CREATION clause
 - IMMEDIATE
 - DEFERRED (default)

```
CREATE TABLE SEG_TAB3(C1 number, C2 number)
  SEGMENT CREATION IMMEDIATE TABLESPACE SEG_TBS;
CREATE TABLE SEG_TAB4(C1 number, C2 number)
  SEGMENT CREATION DEFERRED;
```



Segment creation can be controlled in two ways:

- With the DEFERRED_SEGMENT_CREATION initialization parameter set to TRUE or FALSE. This parameter can be set in the initialization parameter file. You can also control it via the ALTER SESSION or ALTER SYSTEM commands.
- With the SEGMENT CREATION clause of the CREATE TABLE command:
 - SEGMENT CREATION DEFERRED: If specified, segment creation is deferred until the first row is inserted into the table. This is the default behavior.
 - SEGMENT CREATION IMMEDIATE: If specified, segments are materialized during table creation.

This clause takes precedence over the DEFERRED_SEGMENT_CREATION parameter.

It is possible to force the creation of segments for an existing table with the ALTER TABLE ... MOVE command.

It is not possible to directly control the deferred segment creation for dependent objects such as indexes. They inherit this characteristic from their parent object—in this case, the table.

Restrictions and Exceptions

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Segment creation on demand is not for the following:

- IOTs, clustered tables, or other special tables
- Tables in dictionary-managed tablespaces



Segment creation on demand is not supported for IOTs, clustered tables, global temp tables, session-specific temp tables, internal tables, typed tables, AQ tables, SYS-owned tables, external tables, bitmap join indexes, and domain indexes. Tables owned by SYSTEM, PUBLIC, OUTLN, and XDB are also excluded.

Segment creation on demand is not supported for tables created in dictionary-managed tablespaces and for clustered tables. An attempt to do so creates segments.

If you create a table with deferred segment creation on a locally managed tablespace, it has no segments. If at a later time, you migrate the tablespace to be dictionary-managed, any attempt to create segments produces errors. In this case, you must drop the table and re-create it.

Space-Saving Features

- No segments for unusable indexes and index partitions
- Creating an index without a segment:

```
CREATE INDEX test_i1 ON seg_test(c) UNUSABLE
```

- Removing any allocated space for an index:

```
ALTER INDEX test_i UNUSABLE
```

- Creating the segment for an index:

```
ALTER INDEX test_i REBUILD
```



Additional features are implemented in Oracle Database to save space. All UNUSABLE indexes and index partitions are created without a segment. This functionality is completely transparent.

Example: If you have a table with three partitions and a local index, you see three table and three index segments when you query USER_SEGMENTS. If you execute the same query after you move one table partition to a new tablespace, you see three table segments and only two index segments because the unusable one is automatically deleted.

Table Compression Overview

- Reducing storage costs by compressing all data:
 - Basic compression for direct-path insert operations: 10x
 - Advanced row compression for all DML operations: 2–4x

Compression Method	Compression Ratio	CPU Overhead	CREATE and ALTER TABLE Syntax	Typical Applications
Basic table compression	High	Minimal	COMPRESS [BASIC]	DSS
Advanced row compression	High	Minimal	ROW STORE COMPRESS ADVANCED	OLTP, DSS



Oracle Database supports three methods of table compression:

- Basic table compression
- Advanced row compression
- Hybrid columnar compression (with Exadata)

Oracle Corporation recommends compressing all data to reduce storage costs. The Oracle Database server can use table compression to eliminate duplicate values in a data block. For tables with highly redundant data, compression saves disk space and reduces memory use in the database buffer cache. Table compression is transparent to database applications.

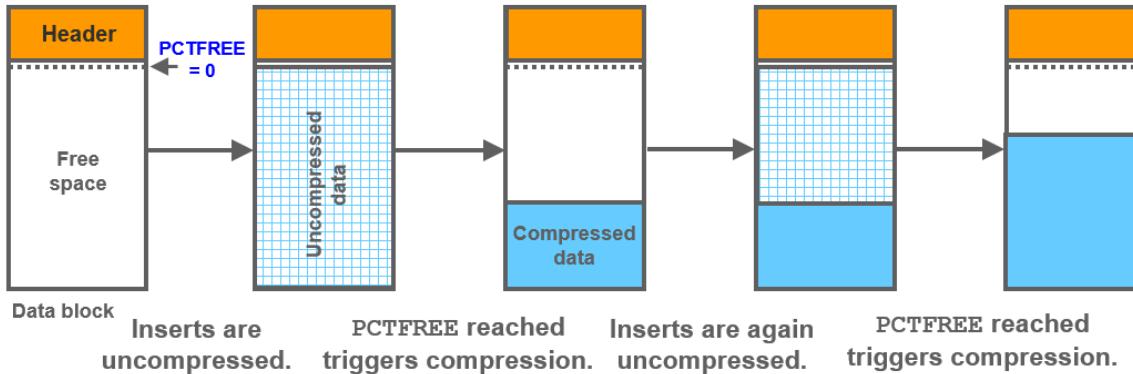
The `TABLE_COMPRESSION` clause is valid only for heap-organized tables. The `COMPRESS` keyword enables table compression. The `NOCOMPRESS` keyword disables table compression. `NOCOMPRESS` is the default.

With basic compression, the Oracle Database server compresses data at the time of performing bulk load using operations such as direct loads or `CREATE TABLE AS SELECT`.

With `ROW STORE COMPRESS ADVANCED`, the Oracle Database server compresses data during all DML operations on the table.

Compression for Direct-Path Insert Operations

- Is enabled with `CREATE TABLE ... COMPRESS BASIC`
- Is recommended for bulk loading data warehouses
- Maximizes contiguous free space in blocks



ORACLE®

Enable basic table compression by using `COMPRESS` or `COMPRESS BASIC`. The Oracle Database server attempts to compress data during the following direct-path insert operations when it is productive to do so.

In earlier releases, this type of compression was called DSS table compression and was enabled using `COMPRESS FOR DIRECT_LOAD OPERATIONS`. This syntax has been deprecated.

Compression eliminates holes created due to deletions and maximizes contiguous free space in blocks.

The diagram above shows you a data block evolution when that block is part of a compressed table. At the start, the block is empty and available for inserts. When you start inserting into this block, data is stored in an uncompressed format (as for uncompressed tables). However, as soon as the block is filled based on the `PCTFREE` setting of the block, the data is automatically compressed, potentially reducing the space it originally occupied.

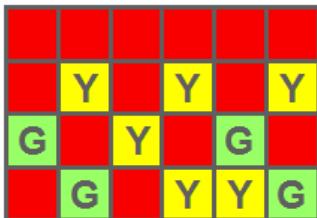
This allows for new uncompressed inserts to take place in the same block, until it is once again filled based on the `PCTFREE` setting. At that point, compression is triggered again to reduce the amount of space used in the block.

Note: Tables with `COMPRESS` or `COMPRESS BASIC` use a `PCTFREE` value of 0 to maximize compression, unless you explicitly set a value for `PCTFREE` clause.

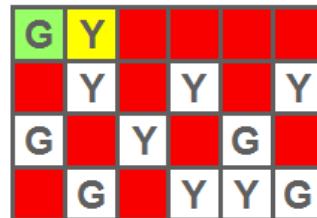
Tables with `ROW STORE COMPRESS ADVANCED` or `NOCOMPRESS` use the `PCTFREE` default value of 10 to maximize compression while still allowing for some future DML changes to the data, unless you override this default explicitly.

Advanced Row Compression for DML Operations

- Is enabled with CREATE TABLE ... ROW STORE COMPRESS ADVANCED
- Is recommended for active OLTP environments



Uncompressed block



OLTP compression with symbol table at the beginning of the block

ORACLE®

Enable advanced row compression by using ROW STORE COMPRESS ADVANCED.

The Oracle database compresses data during all DML operations on the table. This form of compression is recommended for active OLTP environments.

In earlier releases, OLTP table compression was enabled with COMPRESS FOR ALL OPERATIONS and COMPRESS FOR OLTP. This syntax has been deprecated.

With advanced row compression, duplicate values in the rows and columns in a data block are stored once at the beginning of the block in a symbol table. Duplicate values are replaced with a short reference to the symbol table, as shown in the diagram above. Thus, information needed to re-create the uncompressed data is stored in the block.

To illustrate the principle of advanced row compression, the diagram above shows two rectangles. The first red rectangle contains four small green squares labeled "G" and six yellow ones labeled "Y." They represent uncompressed blocks. At the beginning of the second red rectangle, there is only one green square labeled "G" and one yellow "Y" square, representing the symbol table. The second red diagram shows 10 white squares in the same position as the green and yellow ones. They are white because they are now only a reference, not consuming space for duplicate values.

Specifying Table Compression

- You can specify table compression for:
 - An entire heap-organized table
 - A partitioned table (Each partition can have a different type or level of compression.)
 - The storage of a nested table
- You cannot:
 - Specify basic and advanced row compression on tables with more than 255 columns
 - Drop a column if a table is compressed for direct-loads, but you can drop it if the table is advance row compressed

ORACLE®

Table compression has the following restrictions:

- ROW STORE COMPRESS ADVANCED and COMPRESS BASIC are not supported for tables with more than 255 columns.
- You cannot drop a column from a table that is compressed for direct-load operations, although you can set such a column as unused. All the operations of the ALTER TABLE ... *drop_column_clause* are valid for tables that are compressed for OLTP.

Using the Compression Advisor

- Analyzes objects to give an estimate of space savings for different compression methods
- Helps in deciding the correct compression level for an application
- Recommends various strategies for compression
 - Picks the right compression algorithm for a particular data set
 - Sorts on a particular column for increasing the compression ratio
 - Presents tradeoffs between different compression algorithms



The Compression Advisor analyzes database objects and determines the expected compression ratios that can be achieved for each compression level. So it helps you determine the proper compression levels for your application. The advisor recommends various strategies for compression.

A compression advisor, provided by the DBMS_COMPRESSION package, helps you to determine the compression ratio that can be expected for a specified table. The advisor analyzes the objects in the database, discovers the possible compression ratios that could be achieved, and recommends optimal compression levels. In addition to the DBMS_COMPRESSION package, the compression advisor can also be used within the existing advisor framework (with the DBMS_ADVISOR package).

Resolving Space Usage Issues

THESE eKIT MATERIALS ARE FOR YOUR USE ONLY. COPYING OR DISTRIBUTION OF THESE eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



- Resolve space usage issues by:
 - Adding or resizing datafiles
 - Setting AUTOEXTEND to ON
 - Shrinking objects
 - Reducing UNDO_RETENTION
- Check for long-running queries in temporary tablespaces

ORACLE®

Tablespace thresholds are defined either as full or as available space in the tablespace. Critical and warning thresholds are the two thresholds that apply to a tablespace. The DBMS_SERVER_ALERT package contains procedures to set and get the threshold values. When the tablespace limits are reached, an appropriate alert is raised. The threshold is expressed in terms of a percentage of the tablespace size or in remaining bytes free. It is calculated in memory. You can have both a percentage and a byte-based threshold defined for a tablespace. Either or both of them may generate an alert.

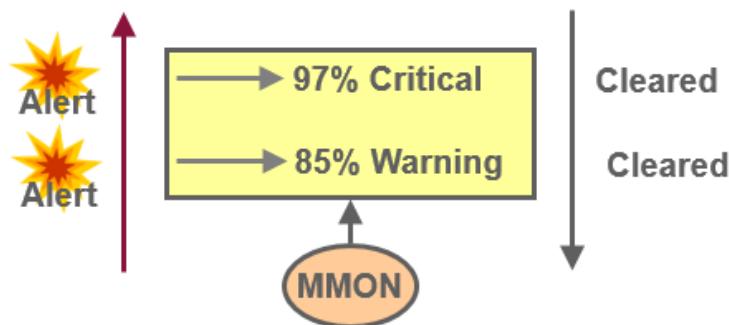
The ideal setting for the warning threshold trigger value results in an alert that is early enough to ensure that there is enough time to resolve the problem before it becomes critical, but late enough so that you are not bothered when space is not a problem.

The alert indicates that the problem can be resolved by doing one or more of the following:

- Adding more space to the tablespace by adding a file or resizing existing files, or making an existing file auto-extendable
- Freeing up space on disks that contain any auto-extendable files
- Shrinking sparse objects in the tablespace

The diagram above shows that the DBA receives a critical alert when the tablespace is 97% full and a warning when the tablespace is 85% full.

Monitoring Tablespace Space Usage



- Read-only and offline tablespaces: Do not set up alerts.
- Temporary tablespace: Threshold corresponds to space currently used by sessions.
- Undo tablespace: Threshold corresponds to space used by active and unexpired extents.
- Auto-extensible files: Threshold is based on the maximum file size.

ORACLE®

The database server tracks space utilization while performing regular space management activities. This information is aggregated by the MMON process. An alert is triggered when the threshold for a tablespace has been reached or cleared.

Alerts should not be flagged on tablespaces that are in read-only mode, or tablespaces that were taken offline, because there is not much to do for them.

In temporary tablespaces, the threshold value has to be defined as a limit on the used space in the tablespace.

For undo tablespaces, an extent is reusable if it does not contain active or unexpired undo. For the computation of threshold violation, the sum of active and unexpired extents is considered as used space.

For tablespaces with auto-extensible files, the thresholds are computed according to the maximum file size you specified, or the maximum OS file size.

The diagram above shows that the MMON process aggregates space utilization information and generates a critical alert when the tablespace is 97% full and a warning when it is 85% full. The alert is cleared after the space usage problem is addressed.

Reclaiming Space by Shrinking Segments

- Shrink is an online and in-place operation.
- It is applicable only to segments residing in ASSM tablespaces.
- Candidate segment types:
 - Heap-organized tables and index-organized tables
 - Indexes
 - Partitions and subpartitions
 - Materialized views and materialized view logs



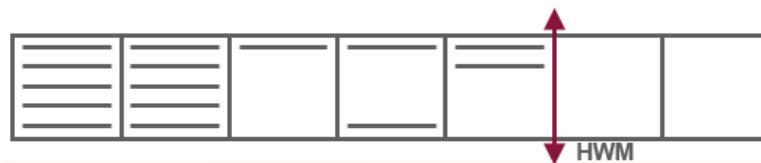
A shrink operation is an online and in-place operation because it does not need extra database space to be executed.

You cannot execute a shrink operation on segments managed by free lists. Segments in automatic segment space-managed tablespaces can be shrunk. However, the following objects stored in ASSM tablespaces cannot be shrunk:

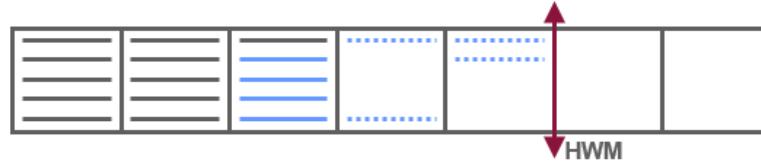
- Tables in clusters
- Tables with `LONG` columns
- Tables with on-commit materialized views
- Tables with `ROWID`-based materialized views
- IOT mapping tables
- Tables with function-based indexes

Because a shrink operation may cause `ROWIDs` to change in heap-organized segments, you must enable row movement on the corresponding segment before executing a shrink operation on that segment. Row movement by default is disabled at segment level.

Shrinking Segments

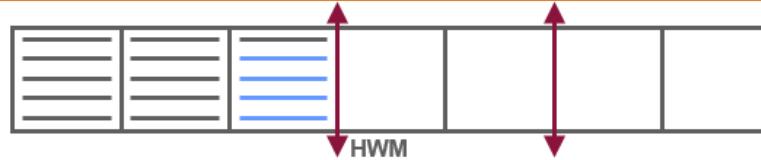


1 `ALTER TABLE employees SHRINK SPACE COMPACT;`



DML operations and queries can be issued during compaction.

2 `ALTER TABLE employees SHRINK SPACE;`



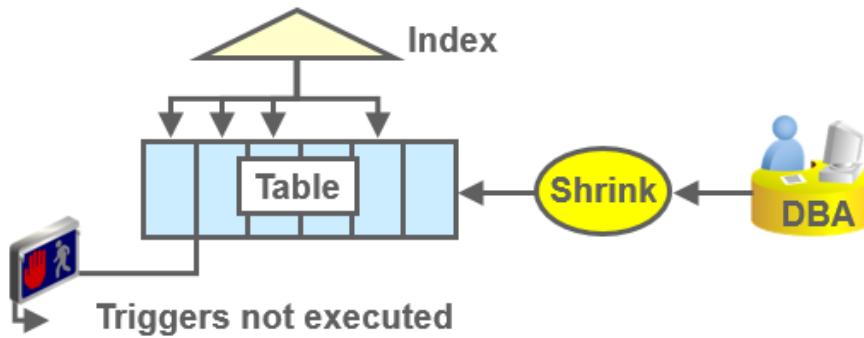
DML operations are blocked when the HWM is adjusted.

ORACLE®

The diagram above describes the two phases of a table shrink operation. Compaction is performed in the first phase. During this phase, rows are moved to the left part of the segment as much as possible. Internally, rows are moved by packets to avoid locking issues. After the rows have been moved, the second phase of the shrink operation is started. During this phase, the high-water mark (HWM) is adjusted and the unused space is released.

The COMPACT clause is useful if you have long-running queries that might span the shrink operation and attempt to read from blocks that have been reclaimed. When you specify the SHRINK SPACE COMPACT clause, the progress of the shrink operation is saved in the bitmap blocks of the corresponding segment. This means that the next time a shrink operation is executed on the same segment, the Oracle Database server remembers what has been done already. You can then reissue the SHRINK SPACE clause without the COMPACT clause during off-peak hours to complete the second phase.

Results of a Shrink Operation



- Improved performance and space utilization
- Indexes maintained
- Triggers not executed
- Number of migrated rows may be reduced.
- Rebuilding secondary indexes on IOTs recommended

ORACLE®

Shrinking a sparsely populated segment improves the performance of scan and DML operations on that segment. This is because there are fewer blocks to look at after the segment has been shrunk. This is especially true for:

- Full table scans (fewer and denser blocks)
- Better index access (fewer I/Os on range ROWID scans due to a more compact tree)

Also, by shrinking sparsely populated segments, you enhance the efficiency of space utilization inside your database because more free space is made available for objects in need.

Index dependency is taken care of during the segment shrink operation. The indexes are in a usable state after shrinking the corresponding table. Therefore, no further maintenance is needed.

The actual shrink operation is handled internally as an `INSERT/DELETE` operation. However, DML triggers are not executed because the data itself is not changed.

As a result of a segment shrink operation, it is possible that the number of migrated rows is reduced. However, you should not always depend on reducing the number of migrated rows after a segment has been shrunk. This is because a segment shrink operation may not touch all the blocks in the segment. Therefore, it is not guaranteed that all the migrated rows are handled.

Note: It is recommended to rebuild secondary indexes on an index-organized table (IOT) after a shrink operation.

Managing Resumable Space Allocation

THESE eKIT MATERIALS ARE FOR YOUR USE ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

A resumable statement:

- Enables you to suspend large operations instead of receiving an error
- Gives you a chance to fix the problem while the operation is suspended, rather than starting over
- Is suspended for the following conditions:
 - Out of space
 - Maximum extents reached
 - Space quota exceeded
- Can be suspended and resumed multiple times.



The Oracle Database server provides a means for suspending, and later resuming, the execution of large database operations in the event of space allocation failures. This enables you to take corrective action instead of the Oracle Database server returning an error to the user. After the error condition is corrected, the suspended operation automatically resumes. This feature is called “resumable space allocation.” The statements that are affected are called “resumable statements.” A statement executes in resumable mode only when the resumable statement feature has been enabled for the system or session.

Suspending a statement automatically results in suspending the transaction. Thus, all transactional resources are held through the suspension and resuming of a SQL statement. When the error condition disappears (for example, as a result of user intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution. A resumable statement is suspended when one of the following conditions occur:

- Out of space condition
- Maximum extents reached condition
- Space quota exceeded condition

A suspension timeout interval is associated with resumable statements. A resumable statement that is suspended for the timeout interval (the default is 2 hours) reactivates itself and returns the exception to the user. A resumable statement can be suspended and resumed multiple times.

Note: A maximum extents reached error happens only with dictionary-managed tablespaces.

Using Resumable Space Allocation

- Queries, DML operations, and certain DDL operations can be resumed if they encounter an out-of-space error.
- A resumable statement can be issued through SQL, PL/SQL, SQL*Loader, and Data Pump utilities, or the Oracle Call Interface (OCI).
- A statement executes in resumable mode only if its session has been enabled by one of the following actions:
 - The RESUMABLE_TIMEOUT initialization parameter is set to a nonzero value.
 - An ALTER SESSION ENABLE RESUMABLE statement is issued.

ORACLE®

Resumable space allocation is possible only when statements are executed within a session that has resumable mode enabled. There are two means of enabling and disabling resumable space allocation:

- Issue the ALTER SESSION ENABLE RESUMABLE command.
- Set the RESUMABLE_TIMEOUT initialization parameter to a nonzero value with an ALTER SESSION or ALTER SYSTEM statement.

When enabling resumable mode for a session or the database, you can specify a timeout period, after which a suspended statement errors out if no intervention has taken place. The RESUMABLE_TIMEOUT initialization parameter indicates the number of seconds before a timeout occurs. You can also specify the timeout period with the following command:

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600
```

The value of TIMEOUT remains in effect until it is changed by another ALTER SESSION ENABLE RESUMABLE statement, it is changed by another means, or the session ends. The default timeout interval when using the ENABLE RESUMABLE TIMEOUT clause to enable resumable mode is 7,200 seconds, or 2 hours.

You can also give a name to resumable statements. Example:

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600
NAME 'multitab insert'
```

~~THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS PROHIBITED.~~

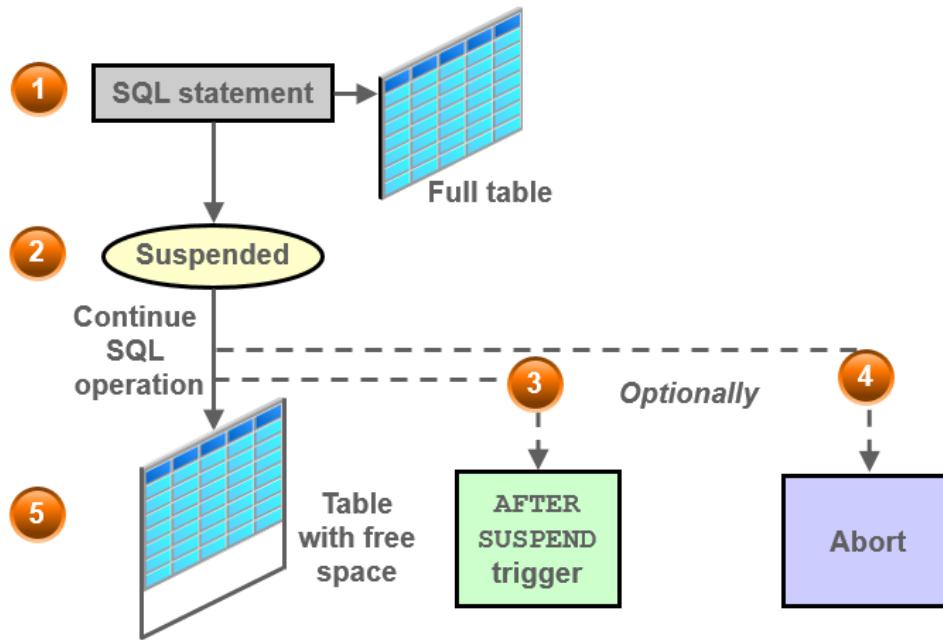
The name of the statement is used to identify the resumable statement in the DBA_RESUMABLE and USER_RESUMABLE views.

To automatically configure resumable statement settings for individual sessions, you can create and register a database-level LOGON trigger that alters a user's session. The trigger issues commands to enable resumable statements for the session, specifies a timeout period, and associates a name with the resumable statements issued by the session.

Because suspended statements can hold up some system resources, users must be granted the RESUMABLE system privilege before they are allowed to enable resumable space allocation and execute resumable statements.

Resuming Suspended Statements

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING OR TRANSFERRING THESE eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



ORACLE®

When a resumable statement is suspended, the error is not raised to the client. In order for corrective action to be taken, the Oracle Database server provides alternative methods for notifying users of the error and for providing information about the circumstances.

The diagram above illustrates the following example:

1. An `INSERT` statement encounters an error saying the table is full.
2. The `INSERT` statement is suspended, and no error is passed to the client.
3. Optionally, an `AFTER SUSPEND` trigger is executed.
4. Optionally, the `SQLERROR` exception is activated to abort the statement.
5. If the statement is not aborted and free space is successfully added to the table, the `INSERT` statement resumes execution.

Possible Actions During Suspension

When a resumable statement encounters a correctable error, the system internally generates the `AFTER SUSPEND` system event. Users can register triggers for this event at both the database and schema level. If a user registers a trigger to handle this system event, the trigger is executed after a SQL statement has been suspended. SQL statements executed within an `AFTER SUSPEND` trigger are always nonresumable and are always autonomous. Transactions started within the trigger use the `SYSTEM` rollback segment. These conditions are imposed to overcome deadlocks and reduce the chance of the trigger experiencing the same error condition as the statement.

Within the trigger code, you can use the `USER_RESUMABLE` or `DBA_RESUMABLE` views, or the

DBMS_RESUMABLE . SPACE_ERROR_INFO function to get information about the resumable statements.

When a resumable statement is suspended:

- The session invoking the statement is put into a wait state. A row is inserted into V\$SESSION_WAIT for the session with the EVENT column containing “statement suspended, wait error to be cleared”.
- An operation-suspended alert is issued on the object that needs additional resources for the suspended statement to complete.

Ending a Suspended Statement

When the error condition is resolved (for example, as a result of DBA intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution and the “resumable session suspended” alert is cleared.

A suspended statement can be forced to activate the SERVERERROR exception by using the DBMS_RESUMABLE . ABORT() procedure. This procedure can be called by a DBA, or by the user who issued the statement. If the suspension timeout interval associated with the resumable statement is reached, the statement aborts automatically and an error is returned to the user.

Summary for Lesson 7

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

In this lesson, you should have learned to:

- Describe how the Oracle Database server automatically manages space
- Save space by using compression
- Proactively monitor and manage tablespace space usage
- Describe segment creation in the Oracle database
- Control deferred segment creation
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation



ORACLE®

Practice 7 Overview

- 7-1: Managing tablespace space
- 7-2: Using compression
- 7-3: Handling resumable space allocation

ORACLE®

Practice 7-1 Managing Tablespace Space

Overview

In this practice you will set a warning threshold and a critical threshold on a tablespace and then test those thresholds. You then create a Segment Advisor task to get recommendations about the current space situation.



Tip:

For problems that cannot be resolved automatically and require DBAs to be notified, such as running out of space, the Oracle Database server provides server-generated alerts. Two alert thresholds are defined by default:

- The *warning threshold* is the limit at which space is beginning to run low.
- The *critical threshold* is a serious limit that warrants your immediate attention.

The database issues alerts at both thresholds. The alerts notify you and often also provide recommendations on how to resolve the reported problem.

Tasks

Set a Warning Threshold

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 `SYSTEM` user and perform the necessary tasks through `SQL*Plus`. See Appendix - [Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB1

...
SQL>
```

3. Execute the package procedure `DBMS_SERVER_ALERT.SET_THRESHOLD` to reset the database-wide threshold values for the Tablespace Space Usage metric. The procedure sets values for the following items:

- `metrics_id`
- `warning_operator`
- `warning_value`
- `critical_operator`

- critical_value
- observation_period
- consecutive_occurrences
- instance_name
- object_type
- object_name

Note: When entering this procedure, you can press return at the end of each line. A > symbol will automatically appear at the beginning of each new line. The > symbol is not part of the code that you enter.

```
SQL> exec DBMS_SERVER_ALERT.SET_THRESHOLD(-
> dbms_server_alert.tablespace_pct_full,-
> NULL,NULL,NULL,NULL,1,1,NULL,-
> dbms_server_alert.object_type_tablespace,NULL) ;
PL/SQL procedure successfully completed.
SQL>
```

4. Check the database-wide threshold values for the Tablespace Space Usage metric.

- Connect to the root container.

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;
Session altered.
```

- Query the WARNING_VALUE and the CRITICAL_VALUE columns in the DBA_THRESHOLDS view. The results show that the warning threshold value is 85 and the critical threshold value is 97.

```
SQL> SELECT warning_value, critical_value
  2  FROM dba_thresholds
  3 WHERE metrics_name='Tablespace Space Usage'
  4 AND object_name IS NULL;

WARNING_VALUE
-----
CRITICAL_VALUE
-----
85
97
SQL>
```

5. In PDB1, create a new tablespace called TBSALERT with a 120MB file called tbsalert.dbf. Make sure that this tablespace is locally managed and uses Automatic Segment Space Management. Do not make it auto-extensible, and do not specify any thresholds for this tablespace.

- Connect to PDB1.

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

b. Create the TBSALERT tablespace.

```
SQL> CREATE TABLESPACE tbsalert
  2  DATAFILE '/u01/app/oracle/oradata/ORCL/PDB1/tbsalert.dbf'
  3  SIZE 120M REUSE LOGGING EXTENT MANAGEMENT LOCAL
  4  SEGMENT SPACE MANAGEMENT AUTO;
```

Tablespace created.

SQL>

6. Query how much free space the TBSALERT tablespace holds.

Note: If you prefer, you can run the script @\$HOME/labs/TBSALERT_free_space.sql instead of typing the code below.

```
SQL> SELECT df.tablespace_name tablespace, fs.bytes free, df.bytes, fs.bytes
 *100/ df.bytes PCT_FREE
  2  FROM dba_data_files df ,dba_free_space fs
  3  WHERE df.tablespace_name = fs.tablespace_name
  4  AND df.tablespace_name = 'TBSALERT';
```

TABLESPACE	FREE	BYTES	PCT_FREE
TBSALERT	124780544	125829120	99.1666667

SQL>

7. Modify the thresholds values for the Tablespace Space Usage metric for the TBSALERT tablespace. Set 55 in the Warning Threshold and 70 in the Critical Threshold.

Note: If you prefer, you can run the script @\$HOME/labs/Tablespace_space_usage_metric_55_70.sql instead of typing the code below.

```
SQL> exec DBMS_SERVER_ALERT.SET_THRESHOLD( metrics_id =>
dbms_server_alert.tablespace_pct_full, warning_operator =>
DBMS_SERVER_ALERT.OPERATOR_GE, warning_value => '55', critical_operator =>
DBMS_SERVER_ALERT.OPERATOR_GE, critical_value => '70', observation_period => 1,
consecutive_occurrences => 1, instance_name => 'ORCL', object_type =>
DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE, object_name => 'TBSALERT' )
```

PL/SQL procedure successfully completed.

SQL>

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

```
SQL> SELECT warning_value, critical_value
  2  FROM dba_thresholds WHERE object_name='TBSALERT';

WARNING_VALUE
-----
CRITICAL_VALUE
-----
55
70
SQL>
```

9. Query the REASON and RESOLUTION columns from the DBA_ALERT_HISTORY view for the TBSALERT tablespace.

```
SQL> SELECT reason, resolution FROM dba_alert_history
  2  WHERE object_name='TBSALERT';

REASON
-----
RESOLUT
-----
Threshold is updated on metrics "Tablespace Space Usage"
cleared
SQL>
```

10. Exit SQL*Plus.

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

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

```
$ $HOME/labs/seg_advsr_setup.sh
...
$
```

12. Check the fullness level of the TBSALERT tablespace to see if the warning level has been reached.

- Start SQL*Plus and connect to PDB1 as the SYSTEM user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus SYSTEM/<password>@PDB1
...
SQL>
```

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

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

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

SQL>
```

- c. Query the number of free bytes that are left in the TBSALERT tablespace. Recall that you created the tablespace with 120MB (125829120 bytes) of space. The query result show that there are 125829120 bytes free and the tablespace is 39% free.

Note: If you prefer, you can run the script @\$HOME/labs/TBSALERT_free_space.sql instead of typing the code below.

```
SQL> SELECT df.tablespace_name tablespace, fs.bytes free, df.bytes ,
  fs.bytes *100/ df.bytes PCT_FREE
  2  FROM dba_data_files df ,dba_free_space fs
  3 WHERE df.tablespace_name = fs.tablespace_name
  4 AND df.tablespace_name = 'TBSALERT';

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

SQL>
```

- d. Wait a few minutes.
- e. Query the DBA_OUTSTANDING_ALERTS view to see if there are any new messages. The REASON column is updated with a message stating that the tablespace is 60 percent full. This message is there because the warning level for the tablespace has been reached. If your result is “no rows selected,” wait a little longer and repeat the query.

```
SQL> SELECT reason FROM dba_outstanding_alerts
  2 WHERE object_name='TBSALERT';

REASON
-----
Tablespace [TBSALERT@PDB1] is [60 percent] full

SQL>
```

Set a Critical Threshold

In this section, you add more data to the TBSALERT tablespace and check the tablespace fullness threshold again.

1. Execute and commit the following INSERT statements.

```
SQL> INSERT INTO hr.employees4 SELECT * FROM hr.employees4;
109568 rows created.

SQL> COMMIT;

Commit complete.

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

109568 rows created.

SQL> COMMIT;

Commit complete.

SQL>
```

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

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

SUM(BYTES)*100/125829120
-----
75
SQL>
```

4. Query the outstanding alerts. The REASON column is updated with a message that states the tablespace is 75 percent full. If your result still displays 60, wait a little longer and repeat the query.

```
SQL> SELECT reason FROM dba_outstanding_alerts WHERE object_name='TBSALERT';

REASON
-----
Tablespace [TBSALERT@PDB1] is [75 percent] full
SQL>
```

5. Delete three tables (and their rows) in the HR schema to try and reduce the space used in the tablespace.

```
SQL> DELETE hr.employees1;
219136 rows deleted.

SQL> COMMIT;

Commit complete.

SQL> DELETE hr.employees2;
219136 rows deleted.

SQL> COMMIT;

Commit complete.

SQL> DELETE hr.employees3;
219136 rows deleted

SQL> COMMIT;

Commit complete.

SQL>
```

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

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

SUM(BYTES)*100/125829120
-----
75
SQL>
```

Create a Segment Advisor Task

1. Create a Segment Advisor task to get recommendations about the current space situation. If you prefer, you can run the @\$HOME/labs/seg_advsr_task.sql script instead of typing the following code.

```

SQL> DECLARE
  2  tname VARCHAR2(128) := 'my_seg_task';
  3  tname_desc  VARCHAR2(128) := 'Get shrink advice for segments in TBSALERT
';
  4  task_id NUMBER;
  5  object_id NUMBER;
  6  objectname VARCHAR2(100);
  7  objecttype VARCHAR2(100);
  8  BEGIN
  9  dbms_advisor.create_task('Segment Advisor', task_id,tname,tname_desc,NULL);
 10 dbms_advisor.create_object(tname,'TABLESPACE','TBSALERT','','','NULL','');
object_id );
11 dbms_advisor.set_task_parameter(tname,'RECOMMEND_ALL','TRUE');
12 END;
13 /

```

PL/SQL procedure successfully completed.

SQL>

2. Execute the task.

```

SQL> DECLARE
  2  tname VARCHAR2(128) := 'my_seg_task';
  3  BEGIN
  4  dbms_advisor.EXECUTE_TASK(tname);
  5  END;
  6  /

```

PL/SQL procedure successfully completed.

SQL>

3. Query the DBA_ADVISOR_TASKS table for recommendations? The recommendation is to get shrink advice for segments stored in the tablespace.

```

SQL> SELECT DESCRIPTION FROM dba_advisor_tasks WHERE TASK_NAME='my_seg_task';

DESCRIPTION
-----
Get shrink advice for segments in TBSALERT
SQL>

```

4. Execute the following query to find out which segments should be shrunk to reclaim space. If you prefer, you can run the @\$HOME/labs/segments_to_shrink.sql script instead of entering the following code. The result shows that the first three segments should be shrunk.

```
SQL> SELECT attr1, attr2, message
  2  FROM dba_advisor_findings f, dba_advisor_objects o
  3  WHERE f.task_name = o.task_name AND f.object_id = o.object_id AND
f.task_name = 'my_seg_task';

ATTR1
-----
ATTR2
-----
MESSAGE
-----
HR
EMPLOYEES1
Perform shrink, estimated savings is 18866176 bytes.

HR
EMPLOYEES2
Perform shrink, estimated savings is 18866176 bytes.

HR
EMPLOYEES3
Perform shrink, estimated savings is 18866176 bytes.

HR
EMPLOYEES4
The free space in the object is less than 10MB.

HR
EMPLOYEES5
The free space in the object is less than 10MB.

SQL>
```

5. Proceed with the SHRINK operation on the HR.EMPLOYEES1, HR.EMPLOYEES2, and HR.EMPLOYEES3 tables.

```
SQL> ALTER TABLE hr.employees1 SHRINK SPACE;
Table altered.

SQL> ALTER TABLE hr.employees2 SHRINK SPACE;
Table altered.

SQL> ALTER TABLE hr.employees3 SHRINK SPACE;
Table altered.

SQL>
```

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

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

SUM(BYTES)*100/125829120
-----
30.15625

SQL>
```

7. Drop the TBSALERT tablespace.

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

Tablespace dropped.

SQL>
```

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

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

Practice 7-2 Using Compression

Overview

In this practice, you will use Advanced Index Compression to reduce the storage for indexes. You use the Compression Advisor, provided by the DBMS_COMPRESSION package, to get detailed space information about compressing the index with different compression levels.

Assumptions

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

Tasks

Complete the following steps.

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. Execute the `$HOME/labs/setup_index.sh` shell script to create an index with low compression on the `HR.TEST` table in `PDB1`.

```
$ $HOME/labs/setup_index.sh

...
$
```

3. Start SQL*Plus and connect to `PDB1` as the `HR` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus hr/<password>@PDB1

...
SQL>
```

4. Query the compression level of the index created on the `HR.test` table. The result indicates that compression is disabled, and therefore, the index is not compressed. The result is formatted for easier viewing.

```
SQL> SELECT index_name, compression FROM user_indexes WHERE index_name =  
'I_TEST';  
  
INDEX_NAME          COMPRESSION  
-----  
I_TEST              DISABLED  
  
SQL>
```

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

```
SQL> SELECT blocks FROM user_segments WHERE segment_name='I_TEST';  
  
BLOCKS  
-----  
1152  
  
SQL>
```

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

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

7. View the different compression levels that exist in your Oracle Database version. To do this, use the cat command to review the pre-defined SQL script that creates the DBMS_COMPRESSION package.

```
$ cat $ORACLE_HOME/rdbms/admin/dbmscomp.sql

...
create or replace package dbms_compression authid current_user is

COMP_NOCOMPRESS          CONSTANT NUMBER := 1;
COMP_ADVANCED              CONSTANT NUMBER := 2;
COMP_QUERY_HIGH            CONSTANT NUMBER := 4;
COMP_QUERY_LOW              CONSTANT NUMBER := 8;
COMP_ARCHIVE_HIGH           CONSTANT NUMBER := 16;
COMP_ARCHIVE_LOW             CONSTANT NUMBER := 32;
COMP_BLOCK                  CONSTANT NUMBER := 64;
COMP_LOB_HIGH                CONSTANT NUMBER := 128;
COMP_LOB_MEDIUM               CONSTANT NUMBER := 256;
COMP_LOB_LOW                  CONSTANT NUMBER := 512;
COMP_INDEX_ADVANCED_HIGH    CONSTANT NUMBER := 1024;
COMP_INDEX_ADVANCED_LOW      CONSTANT NUMBER := 2048;
COMP_BASIC                   CONSTANT NUMBER := 4096;
COMP_INMEMORY_NOCOMPRESS        CONSTANT NUMBER := 8192;
COMP_INMEMORY_DML                 CONSTANT NUMBER := 16384;
COMP_INMEMORY_QUERY_LOW           CONSTANT NUMBER := 32768;
COMP_INMEMORY_QUERY_HIGH          CONSTANT NUMBER := 65536;
COMP_INMEMORY_CAPACITY_LOW         CONSTANT NUMBER := 131072;
COMP_INMEMORY_CAPACITY_HIGH        CONSTANT NUMBER := 262144;

...
$
```

- Start SQL*Plus and connect to PDB1 as the HR user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus hr/<password>@PDB1

...
SQL>
```

- Use the Compression Advisor to get recommendations about the space you would save by compressing the index with the COMP_INDEX_ADVANCED_LOW compression level. To do this, run the following PL/SQL procedure. The slash at the end executes the procedure. The result indicates that the space used by the index would get reduced down to 809 blocks. The Advanced Low Compression ratio equals 1.
If you prefer, you can execute the @\$HOME/labs/Compression_index_low.sql script instead of typing the following code.

```
SQL> set serveroutput on
SQL> DECLARE
      blkcnt_cmp pls_integer;
      blkcnt_ncmp pls_integer;
      row_cmp pls_integer;
      row_ncmp pls_integer;
      cmp_ratio pls_integer;
      comptype_str varchar2(100);
BEGIN
  DBMS_COMPRESSION.GET_COMPRESSION_RATIO
  (
    scratchtbsname => 'USERS',
    ownname => 'HR',
    objname => 'I_TEST',
    subobjname => NULL,
    comptype => dbms_compression.COMP_INDEX_ADVANCED_LOW,
    blkcnt_cmp => blkcnt_cmp,
    blkcnt_ncmp => blkcnt_ncmp,
    row_cmp => row_cmp,
    row_ncmp => row_ncmp,
    cmp_ratio => cmp_ratio,
    comptype_str => comptype_str,
    subset_numrows => dbms_compression.COMP_RATIO_MINROWS,
    objtype => dbms_compression.OBJTYPE_INDEX
  );
  DBMS_OUTPUT.PUT_LINE('Block used by compressed index = ' ||
  blkcnt_cmp);
  DBMS_OUTPUT.PUT_LINE('Block used by uncompressed index = ' ||
  blkcnt_ncmp);
  DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
  DBMS_OUTPUT.PUT_LINE('Compression ratio org = ' || cmp_ratio);
END;
/
Block used by compressed index = 809
Block used by uncompressed index = 1029
Compression type = "Compress Advanced Low"
Compression ratio org = 1
PL/SQL procedure successfully completed.
SQL>
```

10. Use the Compression Advisor again to get recommendations about the space you would save by compressing the index with the COMP_INDEX_ADVANCED_HIGH compression level. To do this, run the following PL/SQL procedure. The result indicates that the space used by the index would get reduced down to 130 blocks. The Advanced High Compression ratio is equal to 8.
If you prefer, you can execute the @\$HOME/labs/Compression_index_high.sql script instead of typing the following code.

```

SQL> set serveroutput on
SQL> DECLARE
      blkcnt_cmp pls_integer;
      blkcnt_ncmp pls_integer;
      row_cmp pls_integer;
      row_ncmp pls_integer;
      cmp_ratio pls_integer;
      comptype_str varchar2(100);
BEGIN
  DBMS_COMPRESSION.GET_COMPRESSION_RATIO
  (
    scratchtbsname => 'USERS',
    ownname => 'HR',
    objname => 'I_TEST',
    subobjname => NULL,
    comptype => dbms_compression.COMP_INDEX_ADVANCED_HIGH,
    blkcnt_cmp => blkcnt_cmp,
    blkcnt_ncmp => blkcnt_ncmp,
    row_cmp => row_cmp,
    row_ncmp => row_ncmp,
    cmp_ratio => cmp_ratio,
    comptype_str => comptype_str,
    subset_numrows => dbms_compression.COMP_RATIO_MINROWS,
    objtype => dbms_compression.OBJTYPE_INDEX
  );
  DBMS_OUTPUT.PUT_LINE('Block used by compressed index = ' ||
    blkcnt_cmp);
  DBMS_OUTPUT.PUT_LINE('Block used by uncompressed index = ' ||
    blkcnt_ncmp);
  DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
  DBMS_OUTPUT.PUT_LINE('Compression ratio org = ' || cmp_ratio);
END;
/
Block used by compressed index = 130
Block used by uncompressed index = 1029
Compression type = "Compress Advanced High"
Compression ratio org = 8

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

11. Question: Based on the previous steps, which compression ratio is the best - the COMP_INDEX_ADVANCED_LOW or COMP_INDEX_ADVANCED_HIGH compression level?
Answer: The Advanced High Compression ratio (8) is much better than the Advanced Low Compression ratio (1). Therefore, you would be inclined to rebuild the index with Advanced High Compression.
12. Rebuild the index with Advanced High Compression.

```
SQL> ALTER INDEX hr.i_test REBUILD COMPRESS ADVANCED HIGH;  
Index altered.  
SQL>
```

13. Query the compression level of the index created on the HR.test table. The result shows that COMPRESSION is equal to ADVANCED HIGH.

```
SQL> SELECT index_name, compression FROM user_indexes WHERE index_name =  
'I_TEST';  
  
INDEX_NAME          COMPRESSION  
-----  
I_TEST              ADVANCED HIGH  
  
SQL>
```

14. Query the space used by the index created on the HR.test table. The space is now 256 blocks.

```
SQL> SELECT blocks FROM user_segments WHERE segment_name='I_TEST';  
  
BLOCKS  
-----  
256  
  
SQL>
```

15. Question: Is it possible to revert back to the initial compression level?
Answer: Yes.

16. Revert back to the initial compression level.

```
SQL> ALTER INDEX hr.i_test REBUILD NOCOMPRESS;  
Index altered.  
SQL>
```

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

```
SQL> SELECT blocks FROM user_segments WHERE segment_name='I_TEST';  
  
BLOCKS  
-----  
1152  
SQL>
```

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

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

Practice 7-3 Handling Resumable Space Allocation

Overview

In this practice, you enable the Resumable Space Allocation feature to avoid situations where a tablespace runs out of space and causes operations to fail, for example, rows cannot be loaded into a table. You will work in two terminal windows (window 1 and window 2).

Tip:

With the Resumable Space Allocation feature:

- Some operations are resumable, but not all. These operations are called resumable statements. `INSERT`, `INSERT INTO` `SELECT`, `UPDATE`, and `DELETE` statements are candidates.
- Some errors are correctable, but not all; for example: out of space condition (`ORA-01653`, `ORA-01654`), maximum extents reached condition (`ORA-01631`, `ORA-01632`), space quota exceeded condition (`ORA-01536`).

Assumptions

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

Tasks

Experience a Table Load Failure in Window 1

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window 1.

```
$ . 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 `SYSTEM` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB1
...
SQL>
```

3. Grant the `PDBADMIN` user the `DBA` role. 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. Execute the \$HOME/labs/CreateINVENTORYTablespace.sql script to create an unpopulated tablespace named INVENTORY. This tablespace is similar to the INVENTORY tablespace you may have worked with in [Practice 6-2 Creating a Tablespace](#).

```
SQL> @$HOME/labs/CreateINVENTORYTablespace.sql  
Tablespace created.  
SQL>
```

6. Execute the \$HOME/labs/CreateTable_X.sql script to create and populate a table named x in the INVENTORY tablespace. As the script runs, notice that rows are being inserted into the table. Part way through the script, you get an error telling you that there is not enough space in the INVENTORY tablespace to insert the remaining rows.

```
SQL> @$HOME/labs/CreateTable_X.sql  
...  
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 from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 -  
64bit Production  
$
```

7. Imagine that the operation in the previous step had lasted 5 hours and that the load had nearly reached its end and other operations were depending on its success.

- Question: Are the rows that did get inserted into the table lost or definitely inserted?
Answer: 2048 rows did get inserted.
- Question: How could this situation be avoided when you do not know how much space is required for a

table to load all its rows?

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

- Start SQL*Plus again and connect to PDB1 as the PDBADMIN user. See [Appendix - Product-Specific Credentials](#) for the password.

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

- Enable resumable mode.

```
SQL> ALTER SESSION ENABLE RESUMABLE;
Session altered.
SQL>
```

- Re-execute the CreateTable_X script. The script is suspended.

```
SQL> @$HOME/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 ;
```

- Question: Why is the script suspended?

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

- Question: Is there any warning message to tell you the load is suspended?

Answer: No. If the script does not execute any further, check the alert log file or the DBA_RESUMABLE view. An operation-suspended alert is issued on the object that needs allocation of resource for the operation to complete.

Research and Fix the Suspended Script in Window 2

- Open another terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL . This window will be referred to as window 2.

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

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

- Start SQL*Plus and connect to PDB1 as SYSTEM.

```
$ sqlplus SYSTEM/<password>@PDB1  
...  
SQL>
```

3. Query the DBA_RESUMABLE view for information about the suspended script. The DBA_RESUMABLE view lists all resumable statements executed in the system. Your times and session information will be different than those shown below.

```
SQL> set pages 100  
SQL> SELECT * FROM dba_resumable;  
  
USER_ID SESSION_ID INSTANCE_ID COORD_INSTANCE_ID COORD_SESSION_ID STATUS  
----- ----- ----- ----- ----- -----  
TIMEOUT START_TIME SUSPEND_TIME RESUME_TIME  
-----  
NAME  
-----  
SQL_TEXT  
-----  
ERROR_NUMBER  
-----  
ERROR_PARAMETER1  
-----  
ERROR_PARAMETER2  
-----  
ERROR_PARAMETER3  
-----  
ERROR_PARAMETER4  
-----  
ERROR_PARAMETER5  
-----  
ERROR_MSG  
-----  
106 17 1 SUSPENDED  
7200 12/08/16 16:36:53 12/08/16 16:36:53  
User PDBADMIN(1  
06), Session 17,  
Instance 1  
INSERT INTO x SELECT * FROM x  
1653  
PDBADMIN  
X  
128  
INVENTORY  
  
ORA-01653: unable to extend table PDBADMIN.X by 128 in tablespace INVENTORY  
SQL>
```

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

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

5. Check the alert log file for information about the suspended script. The log states that the suspension occurred because the database could not extend the table.

```
$ tail -30 /u01/app/oracle/diag/rdbms/orcl/ORCL/trace/alert_ORCL.log
...
2016-12-08T16:36:53.792780+00:00
PDB1(3):statement in resumable session 'User PDBADMIN(106), Session 17, Instance 1' was suspended due to
PDB1(3):      ORA-01653: unable to extend table PDBADMIN.X by 128 in tablespace
INVENTORY
...
$
```

6. Proceed with the appropriate corrective action. Because the INVENTORY tablespace is not autoextensible, you can configure it as autoextensible with a size limit.

- Start SQL*Plus and connect to PDB1 as the SYSTEM user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB1
...
SQL>
```

- Query the DBA_DATA_FILES view to verify whether the INVENTORY tablespace is autoextensible. The result shows that the tablespace is not.

```
SQL> COL file_name FORMAT A50
SQL> SELECT file_name, autoextensible FROM dba_data_files WHERE
tablespace_name='INVENTORY';
FILE_NAME                      AUT
-----
/u01/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF   NO
SQL>
```

- Enable autoextend for the INVENTORY01.DBF data file. Case matters for the data file name.

```
SQL> ALTER DATABASE DATAFILE
'/u01/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF'
  2 AUTOEXTEND ON MAXSIZE 10M;
Database altered.
SQL>
```

- d. Query the DBA_DATA_FILES view again to verify whether the INVENTORY tablespace is autoextensible. The result shows that it is.

```
SQL> COL file_name FORMAT A50
SQL> SELECT file_name, autoextensible
  2  FROM dba_data_files
  3  WHERE tablespace_name='INVENTORY';

FILE_NAME                      AUT
-----
/u01/app/oracle/oradata/ORCL/PDB1/INVENTORY01.DBF YES

SQL>
```

Review the Suspended Session in Window 1

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

```
2048 rows created.

SQL> COMMIT;

Commit complete.

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

2. In window 2, verify that there are no suspended sessions in the system by querying the DBA_RESUMABLE view again.

```
SQL> SELECT * FROM dba_resumable;
no rows selected

SQL>
```

3. Exit SQL*Plus in window 2, and close both terminal windows.

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

8

Managing UNDO Data



ORACLE®

Objectives for Lesson 8

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

After completing this lesson, you should be able to:

- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Enable temporary undo
- Use the Undo Advisor



ORACLE®

Undo Data Overview

Undo data is:

- A record of the action of a transaction
- Captured for every transaction that changes data
- Retained at least until the transaction is ended
- Used to support:
 - Rollback operations
 - Read-consistent queries
 - Oracle Flashback Query, Oracle Flashback Transaction, and Oracle Flashback Table
 - Recovery from failed transactions



The Oracle Database server saves the old value (undo data) when a process changes data in a database. It stores the data as it exists before modifications. Capturing undo data enables you to roll back your uncommitted data. Undo supports read-consistent and flashback queries. Undo can also be used to “rewind” (flash back) transactions and tables.

Read-consistent queries provide results that are consistent with the data as of the time a query started. For a read-consistent query to succeed, the original information must still exist as undo information. If the original data is no longer available, you receive a “Snapshot too old” error (ORA-01555). As long as the undo information is retained, the Oracle Database server can reconstruct data to satisfy read-consistent queries.

Flashback queries purposely ask for a version of the data as it existed at some time in the past. As long as undo information for that past time still exists, flashback queries can complete successfully. Oracle Flashback Transaction uses undo to create compensating transactions, to back out a transaction and its dependent transactions. With Oracle Flashback Table, you can recover a table to a specific point in time.

Undo data is also used to recover from failed transactions. A failed transaction occurs when a user session ends abnormally (possibly because of network errors or a failure on the client computer) before the user decides to commit or roll back the transaction. Failed transactions may also occur when the instance crashes or you issue the SHUTDOWN ABORT command.

In case of a failed transaction, the safest behavior is chosen, and the Oracle Database server reverses all changes made by a user, thereby restoring the original data.

Undo information is retained for all transactions, at least until the transaction is ended by one of the following:

- User undoes a transaction (transaction rolls back).
- User ends a transaction (transaction commits).

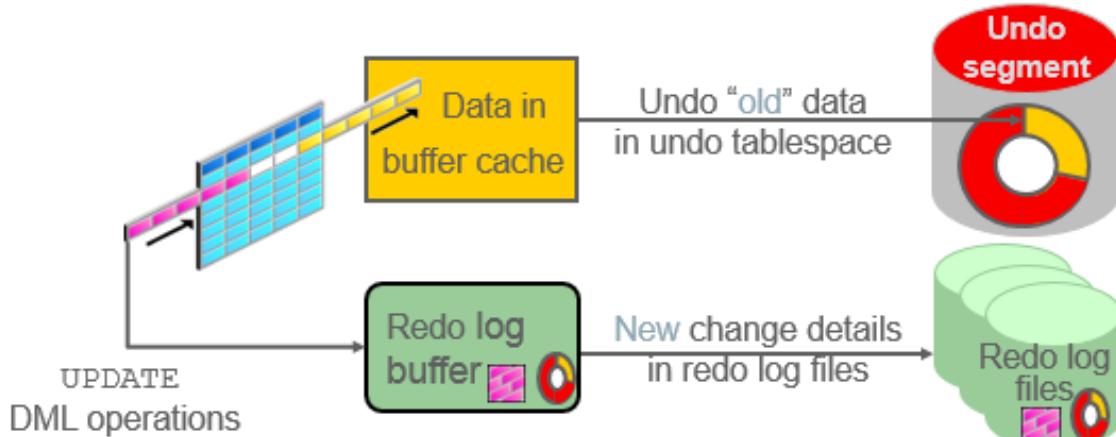
- User executes a DDL statement, such as a CREATE, DROP, RENAME, or ALTER statement. If the current transaction contains any DML statements, the database server first commits the transaction and then executes and commits the DDL as a new transaction.
- User session terminates abnormally (transaction rolls back).
- User session terminates normally with an exit (transaction commits).

The amount of undo data that is retained and the time for which it is retained depend on the amount of database activity and the database configuration.

Note: Oracle Flashback Transaction leverages the online redo logs to mine the appropriate undo SQL for execution. It only uses undo as an artificial time boundary, to determine a redo mining start time for the target transaction, if a transaction start time is not supplied in the flashback transaction invocation.

Transactions and Undo Data

- Each transaction is assigned to only one undo segment.
- An undo segment can service more than one transaction at a time.



ORACLE®

When a transaction starts, it is assigned to an undo segment. Throughout the life of the transaction, when data is changed, the original (before the change) values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the V\$TRANSACTION dynamic performance view.

Undo segments are specialized segments that are automatically created by the database server as needed to support transactions. Like all segments, undo segments are made up of extents, which, in turn, consist of data blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions. Transactions fill extents in their undo segments until a transaction is completed or all space is consumed. If an extent fills up and more space is needed, the transaction acquires that space from the next extent in the segment. After all extents have been consumed, the transaction either wraps around back into the first extent or requests a new extent to be allocated to the undo segment.

Note: Parallel DML and DDL operations can actually cause a transaction to use more than one undo segment. To learn more about parallel DML execution, see the Oracle Database Administrator's Guide.

Storing Undo Information

- Undo information is stored in undo segments, which are stored in an undo tablespace.
- Undo tablespaces:
 - Are used only for undo segments
 - Have special recovery considerations
 - May be associated with only a single instance
 - Require that only one of them be the current writable undo tablespace for a given instance at any given time



Undo segments can exist only in a specialized form of tablespace called an *undo* tablespace. (You cannot create other segment types, such as tables, in the undo tablespace.)

The Database Configuration Assistant (DBCA) automatically creates a smallfile undo tablespace. You can also create a bigfile undo tablespace. However, in a high-volume online transaction processing (OLTP) environment with many short concurrent transactions, contention could occur on the file header. An undo tablespace, stored in multiple data files, can resolve this potential issue.

Although a database may have many undo tablespaces, only one of them at a time can be designated as the current undo tablespace for any instance in the database.

Undo segments are automatically created and always owned by `SYS`. Because the undo segments act as a circular buffer, each segment has a minimum of two extents. The default maximum number of extents depends on the database block size but is very high (32,765 for an 8 KB block size).

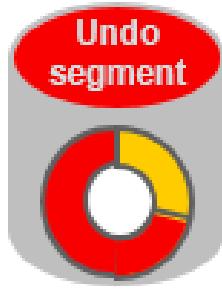
Undo tablespaces are permanent, locally managed tablespaces with automatic extent allocation. They are automatically managed by the database.

Because undo data is required to recover from failed transactions (such as those that may occur when an instance crashes), undo tablespaces can be recovered only while the instance is in the MOUNT state.

Comparing Undo Data and Redo Data

THESE eKIT MATERIALS ARE FOR YOUR UNIVERSITY USE ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

	Undo	Redo
Record of	How to undo a change	How to reproduce a change
Used for	Rollback, read consistency, flashback	Rolling forward of database changes
Stored in	Undo segments	Redo Log files



ORACLE®

Undo data and redo data seem similar at first, but they serve different purposes. Undo data is needed if there is the need to undo a change, and this occurs for read consistency and rollback. Redo data is needed if there is the need to perform the changes again, in cases where they are lost for some reason. Undo block changes are also written to the redo log.

The process of committing entails a verification that the changes in the transaction have been written to the redo log file, which is persistent storage on the disk, as opposed to memory. In addition, the redo log file is typically multiplexed. As a result, there are multiple copies of the redo data on the disk. Although the changes may not yet have been written to the data files where the table's blocks are actually stored, writing to the persistent redo log is enough to guarantee consistency of the database.

Suppose that a power outage occurs just before committed changes have been reflected in the data files. This situation does not cause a problem because the transaction has been committed. When the system starts up again, it is able to roll forward any redo records that are not yet reflected in data files at the time of the outage.

Managing Undo

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASS ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Automatic undo management:

- Fully automated management of undo data and space in a dedicated undo tablespace
- For all sessions
- Self-tuning in AUTOEXTEND tablespaces to satisfy long-running queries
- Self-tuning in fixed-size tablespaces for best retention

DBA tasks in support of Flashback operations:

- Configuring undo retention
- Changing undo tablespace to a fixed size
- Avoiding space and “snapshot too old” errors



The Oracle Database server provides *automatic undo management*, which is a fully automated mechanism for managing undo information and space in a dedicated undo tablespace for all sessions. The system automatically tunes itself to provide the best possible retention of undo information. More precisely, the undo retention period for auto-extending tablespaces is tuned to be slightly longer than the longest-running active query. For fixed-size undo tablespaces, the database dynamically tunes for best possible retention.

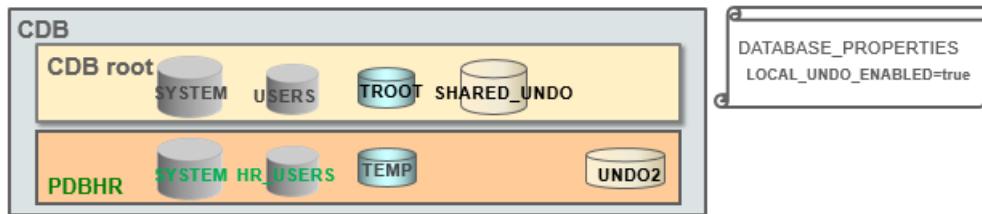
Although by default the Oracle Database server manages undo data and space automatically, you may need to perform some tasks if your database is using Flashback operations. The administration of undo should prevent space errors, the use of too much space, and “Snapshot too old” errors.

Local Undo Mode Versus Shared Undo Mode

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASS ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Two UNDO modes: SHARED versus LOCAL

- There is only one shared UNDO tablespace (in CDB root).
- There can be a local UNDO tablespace in each PDB.



When is local UNDO mode required?

- Hot cloning
- Near-zero downtime PDB relocation

```
SQL> STARTUP UPGRADE;  
SQL> ALTER DATABASE LOCAL UNDO ON;
```

ORACLE®

Using the local UNDO mode is required when cloning a PDB in hot mode or performing a near-zero downtime PDB relocation or refreshing PDBs or using proxy PDBs.

You can set a CDB in local UNDO mode either at CDB creation or by altering the CDB property.

When the database property `LOCAL_UNDO_ENABLED` is `FALSE`, which is the default, there is only one UNDO tablespace that is created in the CDB root and that is shared by all containers.

When `LOCAL_UNDO_ENABLED` is `TRUE`, every container in the CDB uses local undo and each PDB must have its own local UNDO tablespace. To maintain ease of management and provisioning, UNDO tablespace creation happens automatically and does not require any action from the user. When a PDB is opened and an UNDO tablespace is not available, it is automatically created.

Configuring Undo Retention

- UNDO_RETENTION specifies (in seconds) how long already committed undo information is to be retained.
- Set this parameter when:
 - The undo tablespace has the AUTOEXTEND option enabled
 - You want to set undo retention for LOBs
 - You want to guarantee retention



ORACLE®

The UNDO_RETENTION initialization parameter specifies (in seconds) the low threshold value of undo retention. Set the minimum undo retention period for the auto-extending undo tablespace to be as long as the longest expected Flashback operation. For auto-extending undo tablespaces, the system retains undo for at least the time specified in this parameter, and automatically tunes the undo retention period to meet the undo requirements of the queries. But this autotuned retention period may be insufficient for your Flashback operations.

For fixed-size undo tablespaces, the system automatically tunes for the best possible undo retention period on the basis of undo tablespace size and usage history; it ignores UNDO_RETENTION unless retention guarantee is enabled. So for automatic undo management, the UNDO_RETENTION setting is used for the three cases listed in the slide. In cases other than these three, this parameter is ignored.

Categories of Undo

Category	Description
Active: Uncommitted undo information	Supports an active transaction and is never overwritten
Unexpired: Committed undo information	Required to meet the undo retention interval
Expired: Expired undo information	Overwritten when space is required for an active transaction

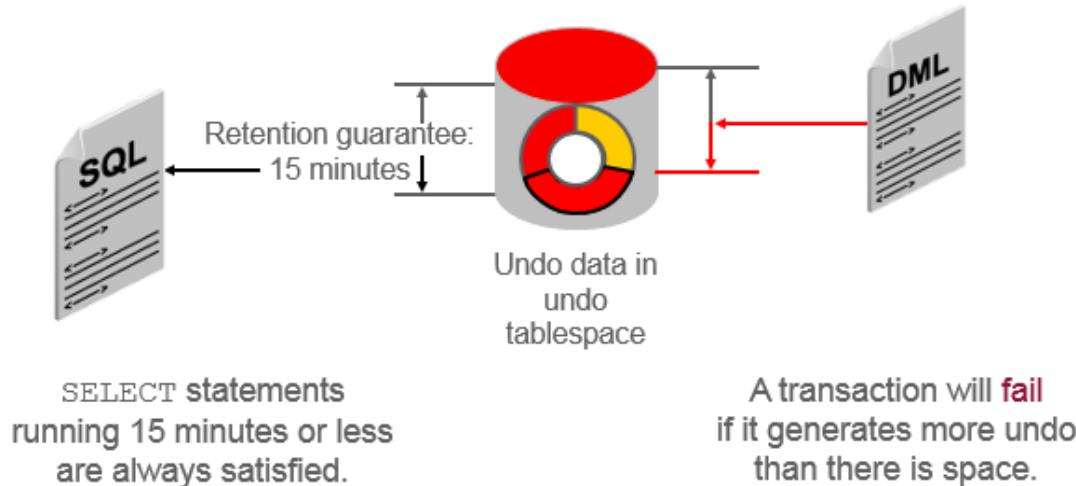


Undo information is divided into three categories:

- Uncommitted undo information (Active): Supports a currently running transaction, and is required if a user wants to roll back or if the transaction has failed. Uncommitted undo information is never overwritten.
- Committed undo information (Unexpired): Is no longer needed to support a running transaction but is still needed to meet the undo retention interval. It is also known as “unexpired” undo information. Committed undo information is retained when possible without causing an active transaction to fail because of lack of space.
- Expired undo information (Expired): Is no longer needed to support a running transaction. Expired undo information is overwritten when space is required by an active transaction.

Guaranteeing Undo Retention

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```



Note: This example is based on an UNDO_RETENTION setting of 900 seconds (15 minutes).

ORACLE®

The default undo behavior is to overwrite the undo information of committed transactions that has not yet expired rather than to allow an active transaction to fail because of lack of undo space.

This behavior can be changed by guaranteeing retention. With guaranteed retention, undo retention settings are enforced even if they cause transactions to fail.

RETENTION GUARANTEE is a tablespace attribute rather than an initialization parameter. This attribute can be changed only with SQL command-line statements.

The syntax to change an undo tablespace to guarantee retention is:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```

To return a guaranteed undo tablespace to its normal setting, use the following command:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;
```

The retention guarantee applies only to undo tablespaces. Attempts to set it on a non-undo tablespace result in the following error:

```
SQL> ALTER TABLESPACE example RETENTION GUARANTEE;
ERROR at line 1:
ORA-30044: 'Retention' can only specified for undo tablespace.
```

Changing an Undo Tablespace to a Fixed Size

- Rationale:
 - Supporting Flashback operations
 - Limiting tablespace growth
- Steps:
 1. Run regular workload.
 2. Self-tuning mechanism establishes minimum required size.
 3. (Optional) Use the Enterprise Manager Cloud Control Undo Advisor, which calculates required size for future growth.
 4. (Optional) Change undo tablespace to a fixed size.



You might have two reasons for changing the undo tablespace to a fixed size: to support Flashback operations (where you expect future use of the undo) or to prevent the tablespace from growing too large.

If you decide to change the undo tablespace to a fixed size, you must choose a large enough size to avoid the following two errors:

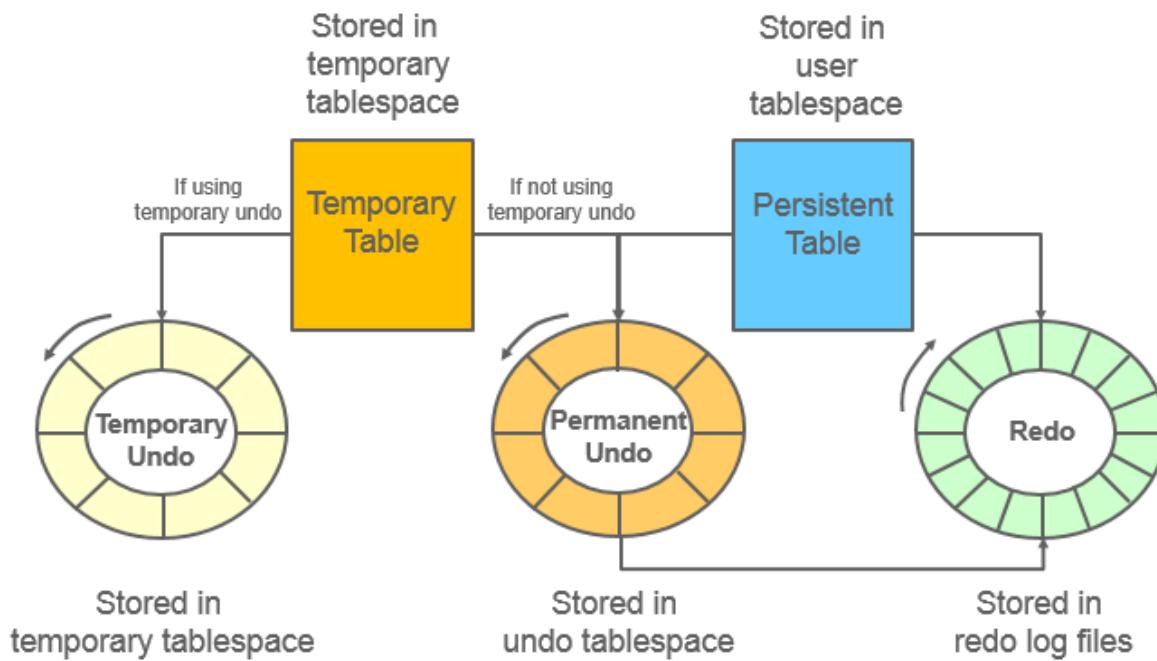
DML failures (because there is not enough space to create the undo for new transactions)

“Snapshot too old” errors (because there was insufficient undo data for read consistency)

Oracle recommends that you run a regular, full workload allowing the undo tablespace to grow to its minimum required size. The automatically gathered statistics include the duration of the longest-running query and the undo generation rate. Computing the minimum undo tablespace size based on these statistics is advisable for a system without Flashback operations, and for a system for which you do not expect longer-running queries in the future.

You can use the Enterprise Manager Cloud Control Undo Advisor to enter your desired duration for the undo period for longer-running queries and flashback.

Temporary Undo Overview



ORACLE®

Temporary tables are widely used as scratch areas for staging intermediate results. This is because changing those tables is much faster than with non-temporary tables. The performance gain is mainly because no redo entries are directly generated for changes on temporary tables. However, the undo for operations on temporary tables (and indices) is still logged to the redo log.

Undo for temporary tables is useful for consistent reads and transaction rollbacks during the life of that temporary object. Beyond this scope the undo is superfluous. Therefore, it need not be persisted in the redo stream. For instance, transaction recovery just discards undo for temporary objects.

Starting with Oracle Database 12c it is possible for undo generated by temporary tables' transactions to be stored in a separate undo stream directly in the temporary tablespace to avoid for that undo to be logged in the redo stream. This mode is called temporary undo.

Note: A temporary undo segment is session private. It stores undo for the changes to temporary tables (temporary objects in general) belonging to the corresponding session.

Temporary Undo Benefits

- Reduces the amount of undo stored in the undo tablespaces
- Reduces the size of the redo log
- Enables DML operations on temporary tables in a physical standby database with the Oracle Active Data Guard option

ORACLE®

Enabling Temporary Undo

- Enable temporary undo for a session:

```
SQL> ALTER SESSION SET temp_undo_enabled = true;
```

- Enable temporary undo for the database instance:

```
SQL> ALTER SYSTEM SET temp_undo_enabled = true;
```

- Temporary undo mode is selected when a session first uses a temporary object.



You can enable temporary undo for a specific session or for the entire database. When you enable temporary undo for a session using an `ALTER SESSION` statement, the session creates temporary undo without affecting other sessions. When you enable temporary undo for the system using an `ALTER SYSTEM` statement, all existing sessions and new sessions create temporary undo.

When a session uses temporary objects for the first time, the current value of the `TEMP_UNDO_ENABLED` initialization parameter is set for the rest of the session. Therefore, if temporary undo is enabled for a session and the session uses temporary objects, then temporary undo cannot be disabled for the session. Similarly, if temporary undo is disabled for a session and the session uses temporary objects, then temporary undo cannot be enabled for the session.

The feature of temporary undo is available for databases with the `COMPATIBLE` initialization parameter set to at least 12.1.0.0.0.

Note: Temporary undo is enabled by default for a physical standby database with the Oracle Active Data Guard option. The `TEMP_UNDO_ENABLED` initialization parameter has no effect on a physical standby database with Active Data Guard option because of the default setting.

Monitoring Temporary Undo

```
SELECT to_char(BEGIN_TIME, 'dd/mm/yy hh24:mi:ss') "BEGIN TIME", TXNCOUNT  
"TXNCNT", MAXCONCURRENCY, UNDOBLKCNT, USCOUNT "USCNT", NOSPACEERRCNT "  
NOSPEERRCNT" FROM V$TEMPUNDOSTAT;  
  
BEGIN TIME          TXNCNT MAXCONCURRENCY UNDOBLKCNT USCNT NOSPEERRCNT  
-----  
...  
19/08/12 22:19:44      0            0            0            0            0  
19/08/12 22:09:44      0            0            0            0            0  
...  
19/08/12 13:09:44      0            0            0            0            0  
19/08/12 12:59:44      3            1            24           1            0  
576 rows selected.  
SQL>
```



V\$TEMPUNDOSTAT shows various statistics related to the temporary undo log for this database instance. It displays a histogram of statistical data to show how the system is working. Each row in the view keeps statistics collected in the instance for a 10-minute interval. The rows are in descending order of the BEGIN_TIME column value. This view contains a total of 576 rows, spanning a 4-day cycle. This view is similar to the V\$UNDOSTAT view.

The example shows you some of the important columns of the V\$TEMPUNDOSTAT view:

- BEGIN_TIME: Identifies the beginning of the time interval
- TXNCOUNT: Total Number of transactions that have bound to temp undo segment within the corresponding time interval
- MAXCONCURRENCY: Highest number of transactions executed concurrently, which modified temporary objects within the corresponding time interval
- UNDOBLKCNT: Total number of temporary undo blocks consumed during the corresponding time interval
- USCOUNT: Temp undo segments created during the corresponding time interval
- NOSPACEERRCNT: Total number of times the error no space left for temporary undo was raised during the corresponding time interval

Note: For more information on V\$TEMPUNDOSTAT, refer to the Oracle Database Reference Guide.

Viewing Undo Information

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

The screenshot shows the Undo Management Details page in Oracle Enterprise Manager Database Express 12c. The top navigation bar includes links for Configuration, Storage, Security, and Performance. Below the navigation is a toolbar with buttons for Change Analysis Parameters, Switch Undo Tablespace, and Create Undo Tablespace. The main content area is divided into two main sections: Undo Summary and Undo Statistics Summary.

Undo Summary

- Undo Setting**: Undo Management is set to auto, and the Low Undo Retention Threshold is 900s.
- Tablespace**: A single undo tablespace named UNDOTBS1 is listed. It has a retention guarantee of No, a size of 70MB (73.8% free), and is auto-extensible with a maximum size of unlimited.
- Errors and Warnings**: Shows 0 Snapshot Too Old Errors, 0 Out of Space Errors, and 0 Unexpired Blocks Stolen.
- Advisor Findings**: Shows Health and Setting both with "No problems".

Undo Statistics Summary

- Analysis Period (Last Day)**: The analysis period from Tuesday Nov 29, 2016 9:29:54 PM to Wednesday Nov 30, 2016 9:26:04 PM had a duration of 23 hours, 56 minutes, 10 seconds. The target undo retention was Required Undo Retention (29 minutes, 9 seconds).
- Undo Retention Analysis**: Required Undo Retention is 29 minutes, 9 seconds, and Best Undo Retention is 210 days, 12 hours, 34 minutes, 32 seconds.
- Undo Statistics**: Shows the following metrics:
 - Undo Generation Rate: 1 KB/s
 - Maximum Undo Used: 55MB
 - Longest SQL: f3yfg50ga0r8n
 - Longest SQL Execution Time: 29 minutes, 9 seconds
 - Transaction Rate: 0 transaction(s) per second
 - Maximum Concurrency: 6

ORACLE

You can view undo information on the Undo Management Details page in Enterprise Manager Database Express. The screenshot above shows the top left two regions of the Undo Management Details page while connected to the container database. The Undo Summary region provides details on undo settings, undo tablespace information, errors and warnings, and the Undo Advisor findings. The Undo Statistics Summary region provides details on the undo analysis period for the last day, output of the undo retention analysis, and undo statistics such as undo generation rate, maximum undo used, longest SQL execution time, transaction rate, and the maximum concurrency.

Viewing Undo Activity



ORACLE®

The Oracle Enterprise Manager Database Express Undo Management Details page includes four additional graphs, as shown on the screenshot above. They include:

- **Undo Generation Rate:** Displays the undo generation (in KB per second)
- **Undo Space Usage:** Shows the use of space in the tablespace with different colors for expired, unexpired, and active extents
- **Steal Activity Breakdown:** Shows the number of attempts to steal expired undo blocks from other undo segments and attempts to obtain undo space by stealing unexpired extents from other transactions
- **Undo Advisor:** Shows the ratio of undo retention times to tablespace sizing needs, along with the current target setting

Practice 8 Overview

- 8-1: Managing Undo Data
- 8-2: Using Local and Shared Undo

Practice 8-1 Managing UNDO Data

Overview

In this practice, you first view your system activity regarding the UNDO feature, and then you configure the ORCL database to support twelve hour retention for flashback operations. Enterprise Manager Database Express (EM Express) enables you to change the UNDO tablespace and perform analysis.

Tip:

Because you use several windows at the same time in this practice, you may find it helpful to change the name of each of them in their banner at the top.

To set a title for a terminal window:

1. In the terminal window's menu, select **Terminal**, and then **Set Title**. A Set Title dialog box is displayed.
2. In the Title box, enter the terminal number, for example, 1, 2, or 3.
3. Click **OK**.

Assumptions

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

Tasks

Start a Workload in the Database

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window 1. Optionally, set the title for the terminal window to 1. See the instructions in the Overview section on how to do that.

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

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

2. Execute the `$HOME/labs/UNDO_setup.sh` shell script to set the UNDO tablespace in the root container to a fixed size.

```
$ $HOME/labs/UNDO_setup.sh

Database altered.
$
```

3. Execute the `$HOME/labs/UNDO_setup_tuning.sh` shell script. This script drops PDB1 and PDB2, recreates them, and creates admin users in each (`admin_pdb1` and `admin_pdb2`). It also creates a user named `oe` in

PDB1 and grants that user the DBA role. Lastly, it creates a tablespace named TBS_APP in PDB1, and creates several tables in the TBS_APP tablespace.

```
$ $HOME/labs/UNDO_setup_tuning.sh  
...  
$
```

4. Open a new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL . This window will be referred to as window 2. Optionally, set the title of the terminal window to 2.

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

5. In window 2, start SQL*Plus and connect to PDB1 as the oe user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus oe/<password>@PDB1  
...  
SQL>
```

6. In window 2, perform two UPDATE transactions in PDB1. The first update may take a minute or two.

```
SQL> UPDATE oe.lineorder set LO_QUANTITY = LO_QUANTITY*10;  
3297032 rows updated.  
  
SQL> UPDATE oe.product_information set min_price = min_price*10;  
287 rows updated.  
SQL>
```

7. Open three more terminal windows and execute the \$HOME/labs/UNDO_loop.sh shell script in each. This script starts a workload in the database. These windows will be referred to as window 3, window 4, and window 5. Optionally, set the title of each terminal window to 3, 4, and 5 respectively.

```
$ $HOME/labs/UNDO_loop.sh  
...  
$
```

Work in EM Express

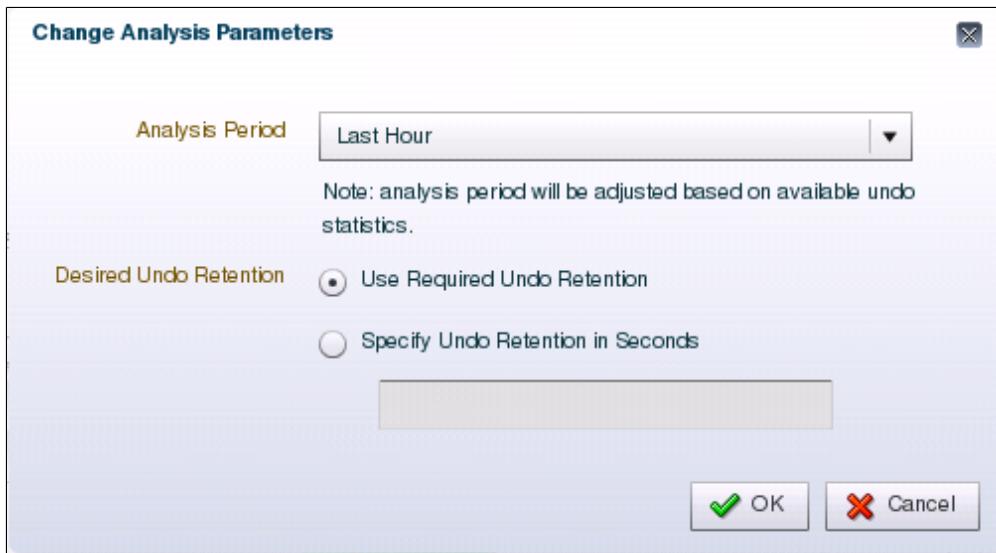
1. Connect to the ORCL CDB with EM Express. To do this:
 - a. Open a Firefox web browser, and enter the URL <https://localhost:5500/em> .
 - b. Enter the user name SYSTEM.
 - c. Enter the password. See [Appendix - Product-Specific Credentials](#) for the password.

- d. Leave the Container Name box empty.
- e. Click **Login**.
- f. If a warning indicates that the connection is untrusted, select the options **I Understand the Risks**, and **Add Security Exception** to continue.

2. Select **Storage**, and then **Undo Management**.

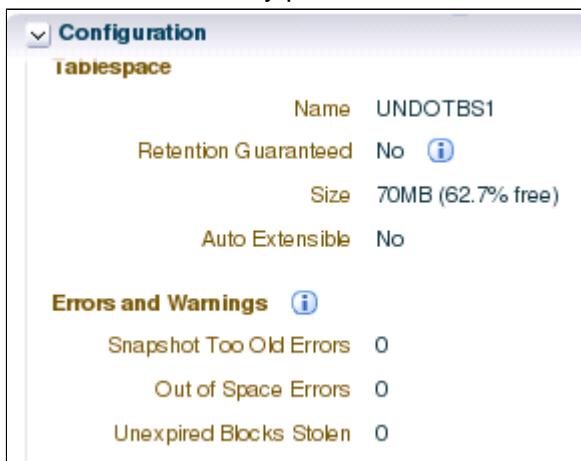
3. Click **Change Analysis Parameters**.

4. In the Analysis Period drop-down list, select **Last Hour**. Leave **Use Required Undo Retention** selected, and click **OK**.



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

6. In the Undo Summary pane, review the UNDO configuration. Your tablespace size may be different.



7. Question: How many errors did this system encounter?

Answer: None. Even if the undo tablespace is currently set to a fixed size, there are no failed transactions.

8. In the Undo Statistics Summary pane, review the information. Your values will be different than those shown below.

Undo Retention Analysis	
Required Undo Retention	9 minutes, 59 seconds
Best Undo Retention	13 hours, 35 minutes, 18 seconds
Undo Statistics	
Undo Generation Rate	1 KB/s
Maximum Undo Used	25MB
Longest SQL	f3yfg50gaOr8n
Longest SQL Execution Time	6 minutes, 14 seconds
Transaction Rate	4 transaction(s) per second

9. Question: What is the duration of the longest running query?
 Answer: In this example, the longest running query is six minutes. Your value will be different.
10. Question: What would be the best undo retention for the past hour statistics collected and the current configuration?
 Answer: In this example, the best undo retention is 13 hours and 35 minutes. Your value will be different.
11. Restart an analysis, keeping the past hour as the period of analysis and defining the Undo Retention to 300 seconds (5 minutes).

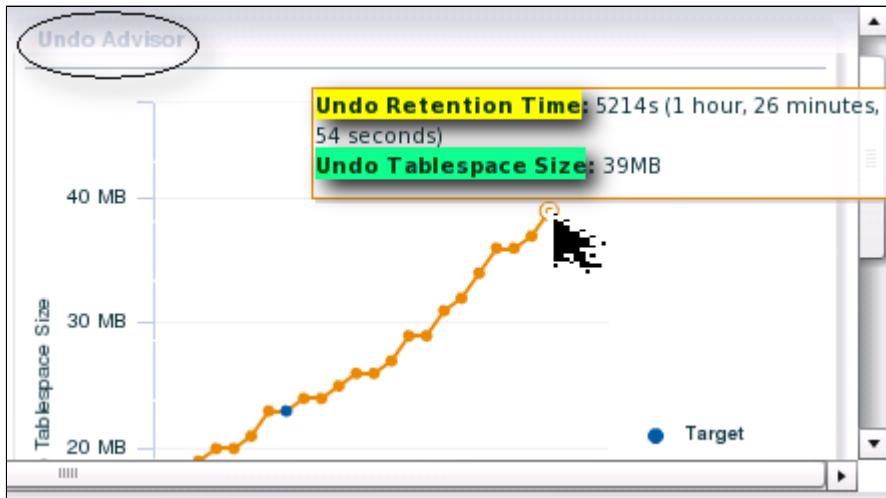
- a. Click **Change Analysis Parameters**.
- b. In the Change Analysis Parameters dialog box, select **Last Hour** from the Analysis Period drop-down list, select the **Specify Undo Retention in Seconds** option, and enter **300**. Click **OK**.
- c. In the Confirmation dialog box, click **OK**.
- d. In the Undo Statistics Summary pane, review the information (particularly the Undo Retention Analysis section).

Undo Retention Analysis	
Required Undo Retention	7 minutes, 56 seconds
Best Undo Retention	8 hours, 1 minute, 20 seconds

- e. Is the best undo retention time the same as before?
 Answer: No, not in this example. With the undo retention configured to 5 minutes, the best undo retention time you may get is 8 hours.
- f. At the bottom of the page, observe the Undo Space Usage during the past hour. There are very few active transactions. But there is a fairly large amount of committed undo information (Unexpired) still needed to meet the undo retention interval. Your graphs will look different than those shown below.



- g. On the right side of the page, the Undo Advisor shows its recommendation. In this example, the undo tablespace size should be at least 39MB for an undo retention time of 1 hour and 26 minutes. The blue point shows the current undo retention value and the UNDO tablespace size. Again, your values will be different.



h. When you have finished analyzing the undo data, log out of EM Express and close the browser window.

12. Close window 1.
13. In window 2, rollback the UPDATE transaction you started earlier. The rollback may take a minute or two to complete.

```
SQL> ROLLBACK;

Rollback complete.
SQL>
```

14. In window 2, exit SQL*Plus and close the terminal window. Click **Close Terminal** in the warning dialog box.

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

15. Close windows 3, 4, and 5, and click **Close Terminal** in each warning dialog box.

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

Practice 8-2 Using Local UNDO

Overview

In this practice, you use SQL*Plus to hot clone PDB1 as PDB4 in the CDB. You experiment with the local UNDO feature while performing the cloning operation. When you are finished, you drop PDB4.

Assumptions

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

Tasks

Attempt to Hot Clone PDB1 with Local UNDO Disabled

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. Execute the `$HOME/labs/HR_PDB1_PDB2_setup.sh` script, which drops, re-creates, and opens PDB1 and PDB2, and populates PDB1 with the HR schema.

```
$ $HOME/labs/HR_PDB1_PDB2_setup.sh

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

3. Execute the `$HOME/labs/setup_undo.sh` shell script which shuts down the database instance, disables the local undo mode in the system, starts up the database instance, and opens all the PDBs.

```
$ $HOME/labs/setup_undo.sh

...
Pluggable database altered.

$
```

4. Create a directory for the new PDB, which we'll name PDB4, under the CDB file location. If the subdirectory is created successfully, no results are returned.

```
$ mkdir $ORACLE_BASE/oradata/ORCL/PDB4  
$
```

5. Start SQL*Plus and connect to the CDB root as the SYS user with the SYSDBA privilege. See [Appendix - Product-Specific Credentials](#) for the password. This session will be referred to as window 1. 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>
```

6. In another terminal window, start a transaction in PDB1.

- a. Open a new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL . This session will be referred to as window 2.

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

- b. Connect to PDB1 as the HR user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus hr/<password>@PDB1  
...  
SQL>
```

- c. Query the commission percent for employee_id 100. The value is null.

```
SQL> SELECT commission_pct FROM hr.employees WHERE employee_id=100;  
  
COMMISSION_PCT  
-----  
  
SQL>
```

- d. Set the commission percent to 10% for employee ID 100. Do not commit the transaction just yet.

```
SQL> UPDATE employees SET commission_pct = .10 WHERE employee_id=100;  
1 row updated.  
SQL>
```

7. In window 1, attempt to clone PDB4 from PDB1 by using the CREATE PLUGGABLE DATABASE statement. You get an error message (ORA-65035).

```
SQL> CREATE PLUGGABLE DATABASE PDB4 FROM PDB1
  2  CREATE_FILE_DEST= '/u01/app/oracle/oradata/ORCL/PDB4' ;

CREATE PLUGGABLE DATABASE PDB4 FROM PDB1
*
ERROR at line 1:
ORA-65035: unable to create pluggable database from PDB1
SQL>
```

8. Find out why the operation failed by researching error message ORA-65035. According to the message, the cloning failed because the PDB being cloned did not have local UNDO enabled.

```
SQL> host oerr ora 65035

65035, 00000, "unable to create pluggable database from %s"
// *Cause: An attempt was made to clone a pluggable database that did not have
//           local undo enabled.
// *Action: Enable local undo for the PDB and and retry the operation.
//
```

SQL>

9. In window 1, verify whether local UNDO is enabled in PDB1. The result shows that the LOCAL_UNDO_ENABLED property is set to FALSE, which means that local UNDO is disabled.

```
SQL> SELECT property_name, property_value
  2  FROM  database_properties
  3 WHERE property_name = 'LOCAL_UNDO_ENABLED';

PROPERTY_NAME      PROPERTY_VALUE
-----
LOCAL_UNDO_ENABLED FALSE

SQL>
```

10. Question: What are your options?

Answer: Either enable local UNDO mode in the CDB, or maintain the shared UNDO mode and set the source PDB1 in READ ONLY mode with no active transaction.

11. You decide to enable local UNDO in the CDB. But before you do so, return to window 2, and rollback the pending transaction in PDB1.

```
SQL> ROLLBACK;

Rollback complete.
SQL>
```

12. In window 2, exit SQL*Plus.

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

Enable Local UNDO Mode (in Window 1)

Enable local UNDO mode in the CDB.

1. Return to window 1 where you are currently connected to the CDB root.
2. Shut down the database instance.

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

3. Start up the instance in UPGRADE mode.

```
SQL> STARTUP UPGRADE  
  
ORACLE instance started.  
  
Total System Global Area 1426063360 bytes  
Fixed Size 8792776 bytes  
Variable Size 486540600 bytes  
Database Buffers 922746880 bytes  
Redo Buffers 7983104 bytes  
Database mounted.  
Database opened.  
SQL>
```

4. Enable local UNDO mode.

```
SQL> ALTER DATABASE LOCAL UNDO ON;  
  
Database altered.  
SQL>
```

5. Restart the database.

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

```
SQL> SHUTDOWN IMMEDIATE

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

- Start up the database instance.

```
SQL> STARTUP

ORACLE instance started.

Total System Global Area 1426063360 bytes
Fixed Size          8792776 bytes
Variable Size       486540600 bytes
Database Buffers   922746880 bytes
Redo Buffers        7983104 bytes
Database mounted.
Database opened.
SQL>
```

- Verify that local UNDO mode is enabled. The query returns a value of TRUE for the LOCAL_UNDO_ENABLED property, which indicates that local UNDO mode is now enabled.

```
SQL> SELECT property_name, property_value
  2  FROM  database_properties
  3 WHERE property_name = 'LOCAL_UNDO_ENABLED';

PROPERTY_NAME      PROPERTY_VALUE
-----
LOCAL_UNDO_ENABLED TRUE
```

Repeat the Hot Cloning Operation

- In window 1, open PDB1.

```
SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN;

Pluggable database altered.
SQL>
```

- In window 2, restart the UPDATE transaction in PDB1.

- a. Start SQL*Plus and connect to PDB1 as the HR user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus hr/<password>@PDB1  
...  
SQL>
```

- b. Set the commission percent to 10% for employee ID 100. Do not commit the transaction just yet.

```
SQL> UPDATE employees SET commission_pct = .10 WHERE employee_id=100;  
1 row updated.  
SQL>
```

3. In window 1, clone PDB1 as PDB4. The cloning is successful.

```
SQL> CREATE PLUGGABLE DATABASE PDB4 FROM PDB1  
  2  CREATE_FILE_DEST = '/u01/app/oracle/oradata/ORCL/PDB4' ;  
  
Pluggable database created.  
  
SQL>
```

4. Open PDB4 so that its open mode is READ WRITE.

```
SQL> ALTER PLUGGABLE DATABASE PDB4 OPEN;  
  
Pluggable database altered.  
SQL>
```

5. In window 2, commit the transaction.

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

6. In window 2, display the new commission for employee ID 100. The value is .1 (10%).

```
SQL> SELECT commission_pct FROM hr.employees WHERE employee_id = 100;  
  
COMMISSION_PCT  
-----  
          .1  
SQL>
```

7. In window 1, verify that the open mode for both PDB1 and PDB4 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        MOUNTED
    5 PDB4        READ WRITE NO

SQL>
```

8. In window 1, connect to PDB4.

```
SQL> ALTER SESSION SET container = PDB4;

Session altered.
SQL>
```

9. In window 1, query the commission percentage of employee ID 100 in PDB4. Notice that the commission percentage did not get updated (it is still null) because the COMMIT statement took place after the clone operation had completed.

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

COMMISSION_PCT
-----
NULL

SQL>
```

Drop PDB4

Continue to work in window 1. In this section, you return to the CDB root and drop the pluggable database.

1. Switch to the CDB root.

```
SQL> ALTER SESSION SET CONTAINER = CDB$ROOT;

Session altered.
SQL>
```

2. Close PDB4.

```
SQL> ALTER PLUGGABLE DATABASE PDB4 CLOSE ;

Pluggable database altered.
SQL>
```

3. Drop PDB4, including its data files, by using the `DROP PLUGGABLE DATABASE` statement.

```
SQL> DROP PLUGGABLE DATABASE PDB4 INCLUDING DATAFILES;
```

```
Pluggable database dropped.  
SQL>
```

4. Exit SQL*Plus in windows 1 and 2, and close both terminal windows.

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

9

Moving Data



ORACLE®

Objectives for Lesson 9

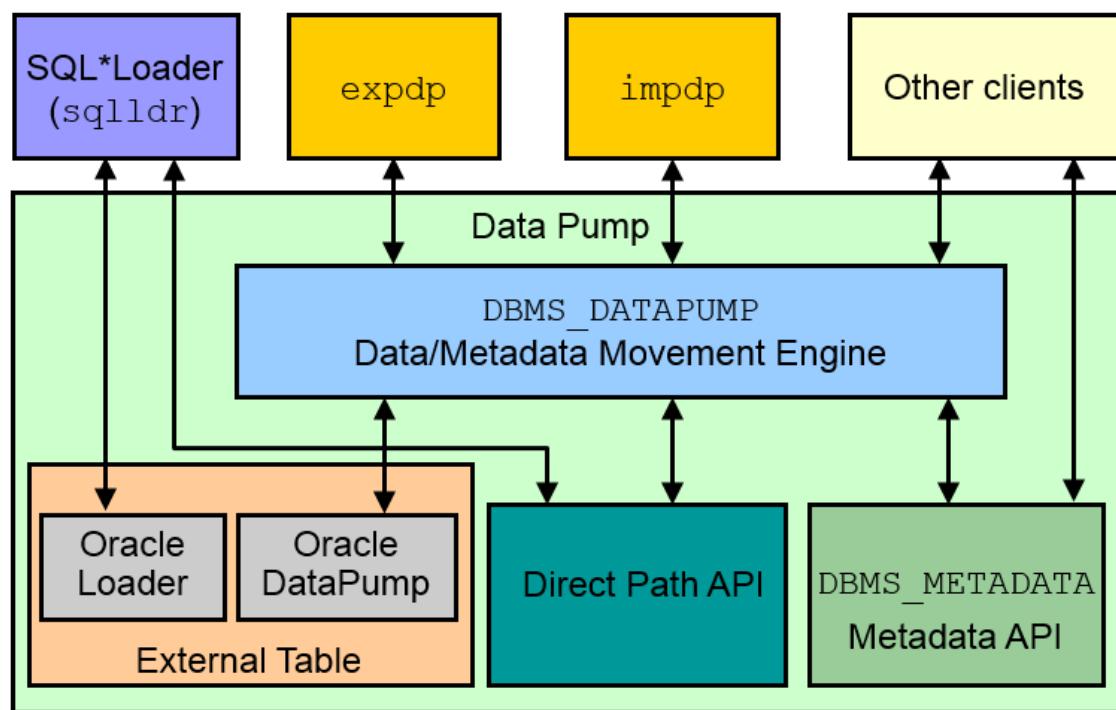
After completing this lesson, you should be able to:

- Describe ways to move data
- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Use external tables to move data via platform-independent files



ORACLE®

Moving Data - General Architecture



ORACLE

Major functional components include:

- **DBMS_DATAPUMP**: Contains the API for high-speed export and import utilities for bulk data and metadata movement
- **Direct Path API (DPAPI)**: Oracle Database supports a Direct Path API interface that minimizes data conversion and parsing at both unload and load time.
- **DBMS_METADATA**: Used by worker processes for all metadata unloading and loading. Database object definitions are stored using XML rather than SQL.
- **External Table**: With the ORACLE_DATAPUMP and ORACLE_LOADER access drivers, you can store data in external tables (that is, in platform-independent files). The SELECT statement reads external tables as though they were stored in an Oracle database.
- **SQL*Loader**: Has been integrated with external tables, providing automatic migration of loader control files to external table access parameters
- **expdp and impdp**: Thin layers that make calls to the DBMS_DATAPUMP package to initiate and monitor Data Pump operations
- **Other clients**: Applications (such as replication, transportable tablespaces, and user applications) that benefit from this infrastructure. SQL*Plus may also be used as a client of DBMS_DATAPUMP for simple status queries against ongoing operations.

Oracle Data Pump Overview

THESE eKIT MATERIALS ARE FOR YOUR CLASS ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

As a server-based facility for high-speed data and metadata movement, Oracle Data Pump:

- Is callable via DBMS_DATAPUMP
- Provides the following tools:
 - expdp and impdp
 - GUI interface in Enterprise Manager Cloud Control
- Provides several data movement methods:
 - Conventional path load
 - Direct path
 - External tables
 - Transportable tablespace
 - Network link support
- Detaches from and re-attaches to long-running jobs
- Restarts Data Pump jobs



Oracle Data Pump enables very high-speed data and metadata loading and unloading of Oracle databases. The Data Pump infrastructure is callable via the DBMS_DATAPUMP PL/SQL package. Thus, custom data movement utilities can be built by using Data Pump.

Oracle Database provides the following tools:

- Command-line export and import clients called expdp and impdp, respectively
- Export and import interface in Enterprise Manager Cloud Control

Data Pump automatically decides the data access methods to use; these can be either direct path or external tables. Data Pump uses direct path load and unload when a table's structure allows it and when maximum single-stream performance is desired. However, if there are clustered tables, encrypted columns, or several other items, Data Pump uses external tables rather than direct path to move the data.

Conventional Path Load is used when Data Pump is not able to load data into a table using either direct path or external tables. The ability to detach from and re-attach to long-running jobs without affecting the job itself enables you to monitor jobs from multiple locations while they are running. All stopped Data Pump jobs can be restarted without loss of data as long as the metadata remains undisturbed. It does not matter whether the job is stopped voluntarily or involuntarily due to a crash.

Oracle Data Pump Benefits

Data Pump offers many benefits and many features, such as:

- Fine-grained object and data selection
- Explicit specification of database version
- Parallel execution
- Network mode in a distributed environment
- Remapping capabilities
- Data sampling and metadata compression
- Compression of data during a Data Pump export
- Security through encryption
- Ability to export XMLType data as CLOBs

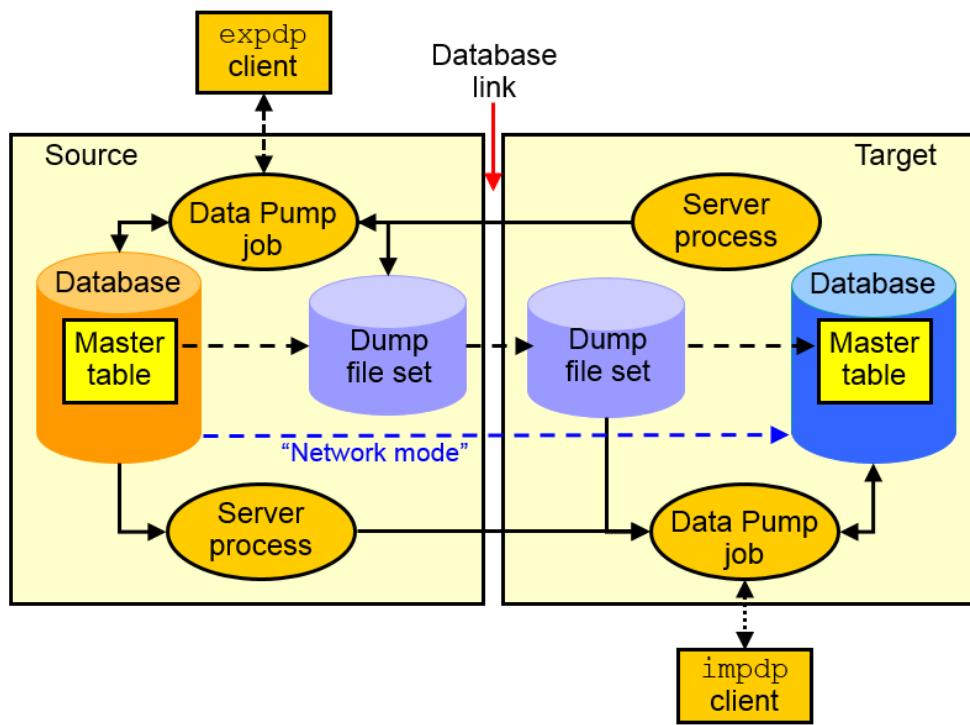


Notes:

- The EXCLUDE, INCLUDE, and CONTENT parameters are used for fine-grained object and data selection.
- You can specify the database version for objects to be moved (using the VERSION parameter) to create a dump file set that is compatible with a previous release of Oracle Database that supports Data Pump.
- You can use the PARALLEL parameter to specify the maximum number of threads of active execution servers operating on behalf of the export job.
- Network mode enables you to export from a remote database directly to a dump file set. This can be done by using a database link to the source system.
- During import, you can change the target data file names, schemas, and tablespaces:
 - Rename tables during an import operation.
 - Remap data as it is being imported into a new database.
- In addition you can specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export. This can be done by specifying the SAMPLE parameter.
- You can use the COMPRESSION parameter to indicate whether the metadata should be compressed in the export dump file so that it consumes less disk space. If you compress the metadata, it is automatically uncompressed during import. You can choose to compress both data and metadata, only data, only metadata, or no data during an export. This feature requires the Oracle Advanced Compression Option.
- You can also specify encryption options to encrypt. Encryption requires the Oracle Advanced Security Option.
 - Both data and metadata, only data, only metadata, no data, or only encrypted columns during an export.
 - With a particular encryption algorithm to use during an export.
 - Using the type of security to use for performing encryption and decryption during an export. For example, perhaps the dump file set will be imported into a different or remote database and it must remain secure

- in transit. Or perhaps the dump file set will be imported on-site using the Oracle Encryption Wallet but it may also need to be imported off-site where the Oracle Encryption Wallet is not available.
- You can also define that XMLType columns are to be exported in uncompressed CLOB format regardless of the XMLType storage format that was defined for them.

Data Pump Export and Import Clients



ORACLE®

Data Pump Export is a utility for unloading data and metadata into a set of operating system files called dump file sets. Data Pump Import is used to load metadata and data stored in an export dump file set into a target system.

The Data Pump API accesses its files on the server rather than on the client.

These utilities can also be used to export from a remote database directly to a dump file set, or to load the target database directly from a source database with no intervening files. This is known as network mode. This mode is particularly useful to export data from a read-only source database.

At the center of every Data Pump operation is the master table, which is a table created in the schema of the user running the Data Pump job. The master table maintains all aspects of the job. The master table is built during a file-based export job and is written to the dump file set as the last step. Conversely, loading the master table into the current user's schema is the first step of a file-based import operation and is used to sequence the creation of all objects imported.

Note: The master table is the key to Data Pump's restart capability in the event of a planned or unplanned stopping of the job. The master table is dropped when the Data Pump job finishes normally.

Data Pump Interfaces and Modes

- Data Pump Export and Import interfaces:
 - Command line
 - Parameter file
 - Interactive command line
 - Enterprise Manager Cloud Control
- Data Pump Export and Import modes:
 - Full
 - Schema
 - Table
 - Tablespace
 - Transportable tablespace
 - Transportable database



You can interact with Data Pump Export and Import by using one of the following interfaces:

- **Command-line interface:** Enables you to specify most of the export parameters directly on the command line
- **Parameter file interface:** Enables you to specify all command-line parameters in a parameter file. The only exception is the PARFILE parameter.
- **Interactive-command interface:** Stops logging to the terminal and displays the export or import prompts, where you can enter various commands. This mode is enabled by pressing `Ctrl + C` during an export operation that is started with the command-line interface or the parameter file interface. Interactive-command mode is also enabled when you attach to an executing or stopped job.
- **Oracle Enterprise Manager Cloud Control:** Select Schema > Database Export/Import. In the menu, select the export or import operation you want to execute.

Data Pump Export and Import provide different modes for unloading and loading different portions of the database. The mode is specified on the command line by using the appropriate parameter. The available modes are listed in the diagram above.

- **FULL=YES:** All data and metadata of the CDB are to be exported. To perform a full export, you must have the `DATAPUMP_EXP_FULL_DATABASE` role.
- **SCHEMAS=hr,oe:** All data and metadata of the schemas are to be exported. This is the default mode for Export. By default, if you do not have the `DATAPUMP_EXP_FULL_DATABASE` role, then only your own schema gets exported. If you have the `DATAPUMP_EXP_FULL_DATABASE` role, then you can specify a list of schemas.
- **TABLES=hr.employees, oe.sales:** Only the specified set of tables, partitions, and their dependent objects are unloaded.

- **TABLESPACES=tbs_app, tbs2:** Only the tables contained in a specified set of tablespaces are unloaded. If a table is unloaded, then its dependent objects are also unloaded. Both object metadata and data are unloaded.
- **TRANSPORT_TABLESPACES=tbs_app, tbs2:** Only object metadata contained in the tablespaces will be exported from the source database into the target database. The data are stored in data files. Because the data files do not get transported with the dump file, they should be copied to the target database before starting the import.
- **TRANSPORTABLE=ALWAYS and FULL=YES:** Both modes used together exports all objects and data necessary to create a complete copy of the database. To import the full transportable database, use TRANSPORT_DATAFILES='datafile1', 'datafile2' parameter to tell import that it is a transportable-mode import and from which data files to get the actual data.

Data Pump Import Transformations

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

You can remap:

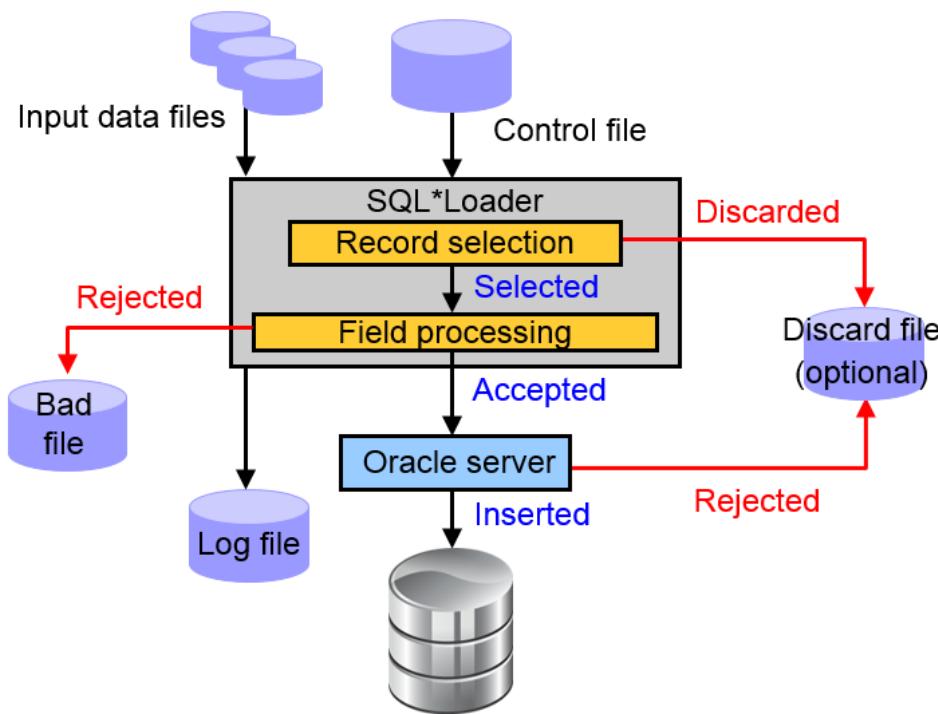
- Data files by using REMAP_DATAFILE
- Tablespaces by using REMAP_TABLESPACE
- Schemas by using REMAP_SCHEMA
- Tables by using REMAP_TABLE
- Data by using REMAP_DATA
- Directory by using REMAP_DIRECTORY



Because object metadata is stored as XML in the dump file set, it is easy to apply transformations when DDL is being formed during import. Data Pump Import supports several transformations:

- REMAP_DATAFILE is useful when moving databases across platforms that have different file-system semantics. It changes the name of the source data file to the target data file name in all SQL statements where the source data file is referenced: CREATE TABLESPACE, CREATE LIBRARY, and CREATE DIRECTORY.
- REMAP_TABLESPACE enables objects to be moved from one tablespace to another.
- REMAP_SCHEMA provides the capability to change object ownership.
- REMAP_TABLE provides the ability to rename entire tables.
- REMAP_DATA provides the ability to remap data as it is being inserted.
- REMAP_DIRECTORY provides the ability to remap directories when you move databases between platforms.

SQL Loader Overview



ORACLE®

SQL*Loader loads data from external files into tables of an Oracle database. It has a powerful data parsing engine that puts little limitation on the format of the data in the data file.

SQL*Loader uses the following files:

- **Input data files:** SQL*Loader reads data from one or more files (or operating system equivalents of files) that are specified in the control file. From SQL*Loader's perspective, the data in the data file is organized as records. A particular data file can be in fixed record format, variable record format, or stream record format. The record format can be specified in the control file with the `INFILE` parameter. If no record format is specified, the default is stream record format.
- **A control file:** The control file is a text file that is written in a language that SQL*Loader understands. The control file indicates to SQL*Loader where to find the data, how to parse and interpret the data, where to insert the data, and so on. Although not precisely defined, a control file can be said to have three sections.
 - The first section contains such session-wide information as the following:
 - Global options, such as the input data file name and records to be skipped
 - `INFILE` clauses to specify where the input data is located
 - Data to be loaded
 - The second section consists of one or more `INTO TABLE` blocks. Each of these blocks contains information about the table (such as the table name and the columns of the table) into which the data is to be loaded.
 - The third section is optional and, if present, contains input data.

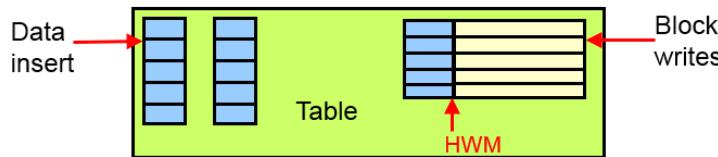
- **A log file:** When SQL*Loader begins execution, it creates a log file. If it cannot create a log file, execution terminates. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load.
- **A discard file:** This file is created only when it is needed and only if you have specified that a discard file should be enabled. The discard file contains records that are filtered out of the load because they do not match any record-selection criteria specified in the control file.
- **A bad file:** The bad file contains records that are rejected, either by SQL*Loader or by the Oracle database. Data file records are rejected by SQL*Loader when the input format is invalid. After a data file record is accepted for processing by SQL*Loader, it is sent to the Oracle database for insertion into a table as a row. If the Oracle database determines that the row is valid, the row is inserted into the table. If the row is determined to be invalid, the record is rejected and SQL*Loader puts it in the bad file.

For more information about SQL*Loader, see the *Oracle Database Utilities* guide.

Loading Methods

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Conventional Load	Direct Path Load
Uses COMMIT	Uses data saves (faster operation)
Always generates redo entries	Generates redo only under specific conditions
Enforces all constraints	Enforces only PRIMARY KEY, UNIQUE, and NOT NULL
Fires INSERT triggers	Does not fire INSERT triggers
Can load into clustered tables	Does not load into clusters
Allows other users to modify tables during load operation	Prevents other users from making changes to tables during load operation
Maintains index entries on each insert	Merges new index entries at the end of the load



ORACLE®

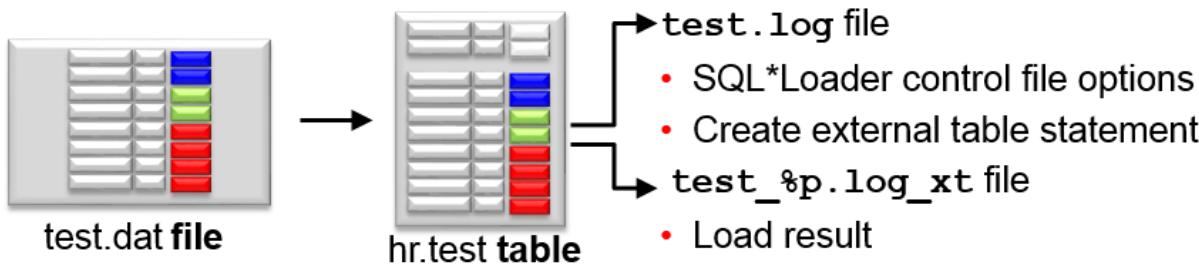
A conventional path load executes SQL INSERT statements to populate tables in an Oracle database. Conventional path loads use SQL processing and a database COMMIT operation for saving data. The insertion of an array of records is followed by a COMMIT operation. Each data load may involve several transactions.

Direct path loads use data saves to format data blocks and write them directly to the data files bypassing the cache layer. This is why the direct path loads are faster than the conventional ones. The following features differentiate a data save from COMMIT:

- During a data save, only full database blocks are written to the database.
- The blocks are written after the high-water mark (HWM) of the table, as shown in the diagram above.
- After a data save, the HWM is moved.
- Internal resources are not released after a data save.
- A data save does not end the transaction.
- If the ROWS parameter is specified, then SQL Loader issues a data save after that many rows are loaded.
- Indexes are not updated at each data save: At the beginning of a direct path load, all indexes on the table are marked unusable. The indexes are updated at the end of the load. If the load is aborted after the indexes were marked unusable, then they will remain unusable.
- The control file option that lets the direct path API handle check constraints is EVALUATE CHECK CONSTRAINTS.

Express Mode

- Specify a table name to initiate an Express Mode load.
- Table columns must be scalar data types (character, number, or datetime).
- A data file can contain only delimited character data.
- SQL*Loader uses table column definitions to determine input data types.
- There is no need to create a control file.



ORACLE®

If you activate SQL*Loader Express Mode, specifying only the `username` and the `TABLE` parameter, it uses default settings for several other parameters. You can override most of the defaults by specifying additional parameters on the command line.

- The `TERMINATED_BY` parameter, which specifies a field terminator
- The `ENCLOSED_BY` parameter, which specifies a field enclosure character
- The `OPTIONALLY_ENCLOSED_BY` parameter, which specifies an optional field enclosure character

The three delimiters can be a multicharacter string.

SQL*Loader Express Mode generates two files. The names of the log files come from the name of the table (by default).

- A log file that includes:
 - The control file output
 - A SQL script for creating the external table and performing the load using a `SQL INSERT /*+ APPEND */ AS SELECT` statement

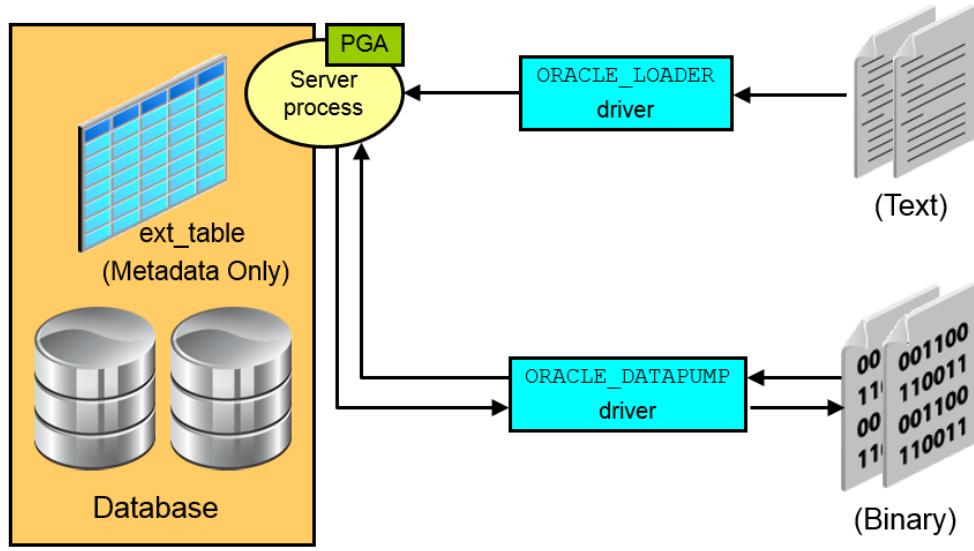
Neither the control file nor the SQL script are used by SQL*Loader Express Mode. They are made available to you in case you want to use them as a starting point to perform operations using regular SQL*Loader or stand-alone external tables.

You can specify direct path load be used instead of external tables with parameter `DIRECT=YES`. You can also specify that conventional path be used instead of external tables with `DIRECT=NO`.

- A log file similar to a SQL*Loader log file that describes the result of the operation. The "%p" represents the process ID of the SQL*Loader process.

External Tables

THESE eKIT MATERIALS ARE FOR YOUR PERSONAL USE ONLY. COPYING OR DISTRIBUTION OF THESE eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



ORACLE®

External tables access data in external sources as if it were in a table in the database. You can connect to the database and create metadata for the external table using DDL. The DDL for an external table consists of two parts:

- One part that describes the Oracle Database column types
- Another part that describes the mapping of the external data to the Oracle Database data columns

An external table does not describe any data that is stored in the database, nor does it describe how data is stored in the external source. Instead, it describes how the external table layer must present the data to the server. It is the responsibility of the access driver and the external table layer to do the necessary transformations required on the data in the external file so that it matches the external table definition. External tables are read only; therefore, no DML operations are possible.

There are two access drivers used with external tables.

- The ORACLE_LOADER access driver can be used only to read table data from an external table and load it into the database. It uses text files as the data source.
- The ORACLE_DATAPUMP access driver can both load table data from an external file into the database and also unload data from the database into an external file. It uses binary files as the external files.

As of Oracle Database 12c Release 2 (12.2.0.1), you can partition data contained in external tables, which allows you to take advantage of the same performance improvements provided when you partition tables stored in a database.

THESE eKIT MATERIALS ARE FOR YOUR PERSONAL USE ONLY. COPYING OR DISTRIBUTION OF THESE eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

External Table Benefits

- Data can be used directly from the external file or loaded into another database.
- External data can be queried and joined directly in parallel with tables residing in the database, without requiring it to be loaded first.
- The results of a complex query can be unloaded to an external file.
- You can combine generated files from different sources for loading purposes.



Summary for Lesson 9

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

In this lesson, you should have learned how to:

- Describe ways to move data
- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Use external tables to move data via platform-independent files



ORACLE®

Practice 9 Overview

- 9-1: Moving data from one PDB to another
- 9-2: Loading data into a PDB from an external file
- 9-3: Querying external tables
- 9-4: Unloading external tables



ORACLE®

Practice 9-1 Moving Data From One PDB to Another

Overview

In this practice, imagine that you configured PDB2 with different optimizer parameter values and you want to test the performance of requests on oe tables within PDB2 to compare it with the performance of the same queries against PDB1. Through trial and error, you export all objects from the oe schema from PDB1 and import them into PDB2 under a new schema named oetest for testing purposes.

Assumptions

You are logged in to VM1 as the oracle user.

Tasks

Export the oe Schema from PDB1 By Using Data Pump Export

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. Execute the \$HOME/labs/DP_setup.sh shell script to create PDB1 and PDB2 with different application tables.

```
$ $HOME/labs/DP_setup.sh

...
$
```

3. Launch Data Pump export under a connection as oe in PDB1 to export all objects belonging to oe. See [Appendix - Product-Specific Credentials](#) for the password. Use the DUMPFILE parameter to specify the location and name of the dump file resulting from the export operation. You will get an error during this operation stating that the file name cannot contain a path specification.

```
$ expdp oe/<password>@PDB1 SCHEMAS=oe
DUMPFILE=/u01/app/oracle/admin/ORCL/dpdump/expoe.dmp

Export: Release 12.2.0.1.0 - Production on Mon Nov 21 16:05:56 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
ORA-39001: invalid argument value
ORA-39000: bad dump file specification
ORA-39088: file name cannot contain a path specification
$
```

4. Question: What does the error message lead you to do?

Answer: Create a logical directory in PDB1. Directory objects are required when you specify file locations for Data Pump because it accesses files on the server rather than on the client. Directory objects are logical structures that represent a physical directory on the server's file system. They contain the location of a specific operating system directory. Directory objects are owned by the `SYS` user. Directory names are unique across the database because all the directories are located in a single name space.

5. 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>
```

6. Create the logical directory into PDB1.

```
SQL> CREATE DIRECTORY dp_for_oe AS '/u01/app/oracle/admin/ORCL/dpdump';
Directory created.
SQL>
```

7. Grant the `oe` user `READ` `WRITE` privileges on the `dp_for_oe` directory.

```
SQL> GRANT read, write ON DIRECTORY dp_for_oe TO oe;
Grant succeeded.
SQL>
```

8. Exit SQL*Plus.

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

9. Retry the Data Pump export.

```
$ expdp oe/<password>@PDB1 SCHEMAS=oe DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp

Export: Release 12.2.0.1.0 - Production on Mon Nov 21 16:20:54 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
Starting "OE"."SYS_EXPORT_SCHEMA_01":  oe/********@pdb1 SCHEMAS=oe
DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp
Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS(TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/COMMENT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
. . exported "OE"."ORDERS"                      12.73 KB    105 rows
. . exported "OE"."ORDER_ITEMS"                  21.01 KB    665 rows
Master table "OE"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
*****
Dump file set for OE.SYS_EXPORT_SCHEMA_01 is:
  /u01/app/oracle/admin/ORCL/dpdump/expoe.dmp
Job "OE"."SYS_EXPORT_SCHEMA_01" successfully completed at Mon Nov 21 16:21:45
2016 elapsed 0 00:00:50
$
```

10. How can you verify that there are objects exported other than tables, for example, dependent objects like constraints, indexes, and sequences?
Answer: Generate a SQL script from the dump file by pretending to import.
11. Generate a SQL script from the dump file. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ impdp oe/<password>@PDB1 SCHEMAS=oe DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp
SQLFILE=oe_SQL

Import: Release 12.2.0.1.0 - Production on Mon Nov 21 16:24:45 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
Master table "OE"."SYS_SQL_FILE_SCHEMA_01" successfully loaded/unloaded
Starting "OE"."SYS_SQL_FILE_SCHEMA_01": oe/********@pdb1 SCHEMAS=oe
DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp SQLFILE=oe_SQL
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS(TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER
Job "OE"."SYS_SQL_FILE_SCHEMA_01" successfully completed at Mon Nov 21 16:24:50
2016 elapsed 0 00:00:04
$
```

12. Review the generated oe_SQL.sql script. All DDL statements are read from the dump file.

```
$ more /u01/app/oracle/admin/ORCL/dpdump/oe_SQL.sql

-- CONNECT OE
ALTER SESSION SET EVENTS '10150 TRACE NAME CONTEXT FOREVER, LEVEL 1';
ALTER SESSION SET EVENTS '10904 TRACE NAME CONTEXT FOREVER, LEVEL 1';
ALTER SESSION SET EVENTS '25475 TRACE NAME CONTEXT FOREVER, LEVEL 1';
ALTER SESSION SET EVENTS '10407 TRACE NAME CONTEXT FOREVER, LEVEL 1';
ALTER SESSION SET EVENTS '10851 TRACE NAME CONTEXT FOREVER, LEVEL 1';
ALTER SESSION SET EVENTS '22830 TRACE NAME CONTEXT FOREVER, LEVEL 192 ';
-- new object type path: SCHEMA_EXPORT/USER
-- CONNECT SYSTEM
CREATE USER "OE" IDENTIFIED BY VALUES 'S:83B6400244803A767307DC6B1FFC68835569E6
1CF89297B52838A1F31668;T:28211B75732FC8F66CDABE661AED86E2F66FDB948D50002FC6DBF2F
CBCBDF12FF47220295029DAFF7619A29A3FC7FBC7F33ABB3F4BB007519D46FA5B4476241D1A40E06
138000ABF7BDA8C1481C88340'
    DEFAULT TABLESPACE "TBS_APP"
    TEMPORARY TABLESPACE "TEMP";
-- new object type path: SCHEMA_EXPORT/SYSTEM_GRANT
GRANT CREATE SESSION TO "OE";
GRANT UNLIMITED TABLESPACE TO "OE";
-- new object type path: SCHEMA_EXPORT/ROLE_GRANT
GRANT "DBA" TO "OE";
-- new object type path: SCHEMA_EXPORT/DEFAULT_ROLE
```

```
ALTER USER "OE" DEFAULT ROLE ALL;
-- new object type path: SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
-- CONNECT OE

BEGIN
sys.dbms_logrep_imp.instantiate_schema(schema_name=>SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA'), export_db_name=>'PDB1.EXAMPLE.COM', inst_scn=>'3638665');
COMMIT;
END;
/
-- new object type path: SCHEMA_EXPORT/SEQUENCE/SEQUENCE
CREATE SEQUENCE "OE"."ORDERS_SEQ" MINVALUE 1 MAXVALUE 999999999999 INCREMENT BY 1 START WITH 10 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE GLOBAL ;
-- new object type path: SCHEMA_EXPORT/TABLE(TABLE)
CREATE TABLE "OE"."ORDERS"
(
    "ORDER_ID" NUMBER(12,0),
    "ORDER_DATE" TIMESTAMP (6) WITH LOCAL TIME ZONE,
    "ORDER_MODE" VARCHAR2(8 BYTE),
    "CUSTOMER_ID" NUMBER(6,0),
    "ORDER_STATUS" NUMBER(2,0),
    "ORDER_TOTAL" NUMBER(12,2),
    "SALES_REP_ID" NUMBER(6,0),
    "PROMOTION_ID" NUMBER(6,0)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "TBS_APP" ;
CREATE TABLE "OE"."ORDER_ITEMS"
(
    "ORDER_ID" NUMBER(12,0),
    "LINE_ITEM_ID" NUMBER(3,0),
    "PRODUCT_ID" NUMBER(6,0),
    "UNIT_PRICE" NUMBER(8,2),
    "QUANTITY" NUMBER(8,0)
) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITTRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "TBS_APP2" ;
-- new object type path: SCHEMA_EXPORT/TABLE/INDEX/INDEX
CREATE INDEX "OE"."I_ORDER_ITEMS" ON "OE"."ORDER_ITEMS" ("ORDER_ID")
PCTFREE 10 INITTRANS 2 MAXTRANS 255
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "TBS_APP" PARALLEL 1 ;

ALTER INDEX "OE"."I_ORDER_ITEMS" NOPARALLEL;
-- new object type path: SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
ALTER TABLE "OE"."ORDERS" ADD PRIMARY KEY ("ORDER_ID")
USING INDEX PCTFREE 10 INITTRANS 2 MAXTRANS 255
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
```

```
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "TBS_APP"  ENABLE;
-- new object type path: SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
-- new object type path: SCHEMA_EXPORT/TABLE/STATISTICS(TABLE_STATISTICS
-- new object type path: SCHEMA_EXPORT/STATISTICS/MARKER
$
```

Import the oe Schema Into PDB2 By Using Data Pump Import

Import the entire oe schema into PDB2 under the new oetest schema.

1. Start SQL*Plus and connect to PDB2 as the SYSTEM user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB2
...
SQL>
```

2. In case the oetest schema already exists in PDB2, drop it first. An error message states that 'OETEST' does not exist.

```
SQL> DROP USER oetest CASCADE;
DROP USER oetest CASCADE
*
ERROR at line 1:
ORA-01918: user 'OETEST' does not exist
SQL>
```

3. Exit SQL*Plus.

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

4. Use the REMAP_SCHEMA parameter to import the entire oe schema into a new oetest schema in PDB2. See [Appendix - Product-Specific Credentials](#) for the password. You will get an error message stating that the directory name for DP_FOR_OE is invalid.

```
$ impdp SYSTEM/<password>@PDB2 REMAP_SCHEMA=oe:oetest DIRECTORY=dp_for_oe  
DUMPFILE=expoe.dmp
```

```
Import: Release 12.2.0.1.0 - Production on Mon Nov 21 16:34:46 2016
```

```
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.
```

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

```
ORA-39002: invalid operation
```

```
ORA-39070: Unable to open the log file.
```

```
ORA-39087: directory name DP_FOR_OE is invalid
```

```
$
```

5. Why does the database report that the directory DP_FOR_OE does not exist when you created that directory in a previous step?

Answer: The directory was created in PDB1, but not in PDB2.

6. Connect to PDB2 and the SYSTEM user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB2  
  
...  
SQL>
```

7. Create the dp_for_oe directory in PDB2.

```
SQL> CREATE DIRECTORY dp_for_oe AS '/u01/app/oracle/admin/ORCL/dpdump';  
  
Directory created.  
SQL>
```

8. Exit SQL*Plus.

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

9. Retry the import operation. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ impdp SYSTEM/<password>@PDB2 REMAP_SCHEMA=oe:oetest DIRECTORY=dp_for_oe
DUMPFILE=expoe.dmp

Import: Release 12.2.0.1.0 - Production on Mon Nov 21 17:02:28 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01": system/********@pdb2
REMAP_SCHEMA=oe:oetest DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp
Processing object type SCHEMA_EXPORT/USER
Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
Processing object type SCHEMA_EXPORT/TABLE/TABLE
ORA-39083: Object type TABLE:"OETEST"."ORDER_ITEMS" failed to create with error:
ORA-00959: tablespace 'TBS_APP2' does not exist

Failing sql is:
CREATE TABLE "OETEST"."ORDER_ITEMS" ( "ORDER_ID" NUMBER(12,0), "LINE_ITEM_ID"
NUMBER(3,0), "PRODUCT_ID" NUMBER(6,0), "UNIT_PRICE" NUMBER(8,2), "QUANTITY"
NUMBER(8,0)) SEGMENT CREATION IMMEDIATE PCTFREE 10 PCTUSED 40 INITTRANS 1
MAXTRANS 255 NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1
MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT) TABLESPACE "TBS_APP2"

Processing object type SCHEMA_EXPORT/TABLE(TABLE_DATA
. . imported "OETEST"."ORDERS" 12.73 KB 105 rows
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
ORA-39112: Dependent object type INDEX:"OETEST"."I_ORDER_ITEMS" skipped, base
object type TABLE:"OETEST"."ORDER_ITEMS" creation failed

Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS/TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER
Job "SYSTEM"."SYS_IMPORT_FULL_01" completed with 2 error(s) at Mon Nov 21
17:02:36 2016 elapsed 0 00:00:08
$
```

10. Question: Did the Data Pump import complete the import?

Answer: Not completely. Data Pump imported only the objects that it could process without any error.

11. Question: Which objects were not imported?

Answer: Data Pump could not import the ORDER_ITEMS table because this table requires the TBS_APP2 tablespace, which does not exist in PDB2. Therefore, the dependent objects of this table could not be imported as well, like the index.

12. Create the TBS_APP2 tablespace in PDB2.

- a. Start SQL*Plus and connect to PDB2 and the SYSTEM user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus SYSTEM/<password>@PDB2  
...  
SQL>
```

- b. Issue the CREATE TABLESPACE command to create the TBS_APP2 tablespace in PDB2.

```
SQL> CREATE TABLESPACE tbs_app2 DATAFILE  
'/u01/app/oracle/oradata/ORCL/pdb2/tbs_app02.dbf' SIZE 100m;  
  
Tablespace created.  
SQL>
```

- c. Exit SQL*Plus.

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

13. Retry the import operation. See Appendix - Product-Specific Credentials for the password.

```
$ impdp SYSTEM/<password>@PDB2 REMAP_SCHEMA=oe:oetest DIRECTORY=dp_for_oe
DUMPFILE=expoe.dmp

Import: Release 12.2.0.1.0 - Production on Thu Nov 24 12:09:26 2016
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.
Connected to: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit
Production
Master table "SYSTEM"."SYS_IMPORT_FULL_01" successfully loaded/unloaded
Starting "SYSTEM"."SYS_IMPORT_FULL_01": system/********@pdb2
REMAP_SCHEMA=oe:oetest DIRECTORY=dp_for_oe DUMPFILE=expoe.dmp
Processing object type SCHEMA_EXPORT/USER
ORA-31684: Object type USER:"OETEST" already exists

Processing object type SCHEMA_EXPORT/SYSTEM_GRANT
Processing object type SCHEMA_EXPORT/ROLE_GRANT
Processing object type SCHEMA_EXPORT/DEFAULT_ROLE
Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA
Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE
ORA-31684: Object type SEQUENCE:"OETEST"."ORDERS_SEQ" already exists

Processing object type SCHEMA_EXPORT/TABLE/TABLE
ORA-39151: Table "OETEST"."ORDERS" exists. All dependent metadata and data will
be skipped due to table_exists_action of skip

Processing object type SCHEMA_EXPORT/TABLE(TABLE_DATA
. . imported "OETEST"."ORDER_ITEMS" 21.01 KB 665 rows
Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX
Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS
Processing object type SCHEMA_EXPORT/TABLE/STATISTICS(TABLE_STATISTICS
Processing object type SCHEMA_EXPORT/STATISTICS/MARKER
Job "SYSTEM"."SYS_IMPORT_FULL_01" completed with 3 error(s) at Thu Nov 24
12:09:34 2016 elapsed 0 00:00:07
$
```

14. Question: Are the errors true errors?

Answer: The errors are normal errors stating that objects were already created during the previous import operation.

Verify the oetest Schema in PDB2

Verify that the new oetest schema exists in PDB2.

1. Start SQL*Plus and connect to PDB2 as the oetest user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus oetest/<password>@PDB2
...
SQL>
```

2. View the list of tables to which the oetest user has access.

```
SQL> SELECT table_name FROM user_tables;

TABLE_NAME
-----
ORDERS
ORDER_ITEMS
SQL>
```

3. Query the number of rows in the ORDER_ITEMS table. The results show that there are 665 rows.

```
SQL> SELECT count(*) FROM order_items;

COUNT(*)
-----
665
SQL>
```

4. List the indexes to which the oetest user has access.

```
SQL> SELECT index_name FROM user_indexes;

INDEX_NAME
-----
SYS_C007392
I_ORDER_ITEMS
SQL>
```

5. List the sequences to which the oetest user has access.

```
SQL> SELECT sequence_name FROM user_sequences;

SEQUENCE_NAME
-----
ORDERS_SEQ
SQL>
```

6. Exit SQL*Plus.

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

7. Question: How could you have imported the oe schema from PDB1 to PDB2 in one single operation?

Answer: The data could be imported from PDB1 by using a valid database link and written directly back to the connected PDB2. The Data Pump import uses the NETWORK_LINK parameter to define the database link used to access the database from which to import the data.

[Import the oe Schema into PDB2 Via a Database Link](#)

1. Start SQL*Plus and connect to PDB2 as the SYSTEM user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB2  
...  
SQL>
```

2. Create a database link in the destination PDB (PDB2) that will connect to the source PDB (PDB1). See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CREATE DATABASE LINK link_pdb1  
  2  CONNECT TO SYSTEM IDENTIFIED BY <password> USING 'PDB1';  
  
Database link created.  
SQL>
```

3. Drop the target user created in the previous import operation.

```
SQL> DROP USER oetest CASCADE;  
  
User dropped.  
SQL>
```

4. Exit SQL*Plus.

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

5. Use the NETWORK_LINK parameter to initiate an import via a database link. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ impdp SYSTEM/<password>@PDB2 SCHEMAS=oe REMAP_SCHEMA=oe:oetest
NETWORK_LINK=link_pdb1
```

Import: Release 12.2.0.1.0 - Production on Thu Nov 24 14:27:27 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

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

Starting "SYSTEM"."SYS_IMPORT_SCHEMA_01": system/********@pdb2 SCHEMAS=oe REMAP_SCHEMA=oe:oetest NETWORK_LINK=link_pdb1

Estimate in progress using BLOCKS method...

Processing object type SCHEMA_EXPORT/TABLE/TABLE_DATA

Total estimation using BLOCKS method: 128 KB

Processing object type SCHEMA_EXPORT/USER

Processing object type SCHEMA_EXPORT/SYSTEM_GRANT

Processing object type SCHEMA_EXPORT/ROLE_GRANT

Processing object type SCHEMA_EXPORT/DEFAULT_ROLE

Processing object type SCHEMA_EXPORT/PRE_SCHEMA/PROCACT_SCHEMA

Processing object type SCHEMA_EXPORT/SEQUENCE/SEQUENCE

Processing object type SCHEMA_EXPORT/TABLE/TABLE

. . . imported "OETEST"."ORDERS" 105 rows

. . . imported "OETEST"."ORDER_ITEMS" 665 rows

Processing object type SCHEMA_EXPORT/TABLE/INDEX/INDEX

Processing object type SCHEMA_EXPORT/TABLE/CONSTRAINT/CONSTRAINT

Processing object type SCHEMA_EXPORT/TABLE/INDEX/STATISTICS/INDEX_STATISTICS

Processing object type SCHEMA_EXPORT/TABLE/STATISTICS(TABLE_STATISTICS)

Processing object type SCHEMA_EXPORT/STATISTICS/MARKER

Job "SYSTEM"."SYS_IMPORT_SCHEMA_01" successfully completed at Thu Nov 24

14:32:21 2016 elapsed 0 00:00:57

\$

6. Verify that the oe schema was imported as oetest into PDB2.

- Start SQL*Plus and connect to PDB2 as the oetest user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus oetest/<password>@PDB2
```

...

SQL>

- View the list of tables to which the oetest user has access.

```
SQL> SELECT table_name FROM user_tables;
```

TABLE_NAME

ORDER_ITEMS

ORDERS

SQL>

- Query the number of rows in the ORDER_ITEMS table. The table has 665 rows.

```
SQL> SELECT count(*) FROM order_items;  
      COUNT( * )  
-----  
       665  
SQL>
```

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

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

7. Question: What are the advantages and drawbacks of this type of Data Pump import?

Answer: There are no dump files involved. If an import operation is performed over an unencrypted network link, then all data is imported as clear text even if it is encrypted in the database.

Practice 9-2 Loading Data into a PDB from an External File

Overview

In this practice, you use SQL*Loader to perform the following load operations:

1. Load data into the SH.PRODUCTS table in PDB2 by using SQL*Loader in express mode. Data and control files are provided.
2. Load data into the SH.INVENTORIES table in PDB2 by using SQL*Loader in conventional mode.
3. Load data into the SH.INVENTORIES table in PDB2 by using SQL*Loader in direct mode.

Assumptions

You are logged in to VM1 as the oracle user.

Tasks

Load Data by Using SQL*Loader in Express Mode

As the sh user, use SQL*Loader in Express Mode to load data into the SH.PRODUCTS table in PDB2 from the \$HOME/labs/products.dat data file.

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. If you did not complete Practice 9-1 Moving Data From One PDB to Another, execute the \$HOME/labs/DP_setup.sh shell script. This script creates PDB1 and PDB2 with different application tables.

```
$ $HOME/labs/DP_setup.sh

...
$
```

3. View the products.dat file to learn about its structure before going further.

```
$ cat /home/oracle/labs/products.dat  
  
4001,ENG,Door,Outdoor  
4002,FRE,Porte,Porte exterieure  
4003,SPA,Puerta,Puerta exterior  
4004,GER,Tur,Auberliche Tur  
5001,ENG,Shutter,Outdoor shutter  
5002,FRE,Volet,Volet exterieur  
5003,SPA,Obturador,Obturador exterior  
5004,GER,Fenster, Fensterladen  
$
```

4. Start SQL*Plus and connect to PDB2 as the sh user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus sh/ <password> @PDB2  
  
...  
SQL>
```

5. Count the number of products already loaded into the SH.PRODUCTS table. The results indicate that there are six rows in the table, and therefore, six products.

```
SQL> select count(*) from sh.products;  
  
COUNT( * )  
-----  
       6  
SQL>
```

6. Exit SQL*Plus.

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

7. Change to the directory /home/oracle/labs.

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

8. Start SQL*Loader, connect to PDB2 as the sh user, and load the records from the products.dat file into the SH.PRODUCTS table in PDB2. See [Appendix - Product-Specific Credentials](#) for the password. The results show that eight rows were successfully loaded.

```
$ sqlldr sh/<password>@PDB2 TABLE=products  
  
SQL*Loader: Release 12.2.0.1.0 - Production on Mon Nov 21 18:39:11 2016  
  
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.  
  
Express Mode Load, Table: PRODUCTS  
Path used:          External Table, DEGREE_OF_PARALLELISM=AUTO  
  
Table PRODUCTS:  
  8 Rows successfully loaded.  
  
Check the log files:  
  products.log  
  products_%p.log_xt  
for more information about the load.  
$
```

9. Start SQL*Plus and connect to PDB2 as the sh user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus sh/<password>@PDB2  
...  
SQL>
```

10. Verify that the table is loaded with the 8 records from the products.dat file. The results show that the records were loaded.

```
SQL> SELECT * FROM products;

PRODUCT_ID COU LABEL          DETAILED_LABEL
----- -----
 1001 ENG Shutter1    Outdoor shutter1
 1002 FRE Portel     Porte exterieure1
 1003 SPA Puerta1    Puerta exterior1
 1004 GER Tur1       Auberliche Tur1
 1005 FRE Volet1     Volet exterieur1
 1007 GER Fenster1   Fensterladen1
 4001 ENG Door       Outdoor
 4002 FRE Porte      Porte exterieure
 4003 SPA Puerta     Puerta exterior
 4004 GER Tur        Auberliche Tur
 5001 ENG Shutter    Outdoor shutter
 5002 FRE Volet     Volet exterieur
 5003 SPA Obturador Obturador exterior
 5004 GER Fenster    Fensterladen

14 rows selected.
SQL>
```

11. Exit SQL*Plus.

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

12. View the products.log file.

```
$ cat products.log

SQL*Loader: Release 12.2.0.1.0 - Production on Mon Nov 21 18:39:11 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Express Mode Load, Table: PRODUCTS
  Data File:      products.dat
  Bad File:      products_%p.bad
  Discard File:  none specified
  (Allow all discards)

  Number to load: ALL
  Number to skip: 0
  Errors allowed: 50
  Continuation:   none specified
  Path used:      External Table

  Table PRODUCTS, loaded from every logical record.
  Insert option in effect for this table: APPEND
```

Column Name	Position	Len	Term	Encl	Datatype
PRODUCT_ID	FIRST	*	,		CHARACTER
COUNTRY	NEXT	*	,		CHARACTER
LABEL	NEXT	*	,		CHARACTER
DETAILED_LABEL	NEXT	*	,		CHARACTER

```

Generated control file for possible reuse:
OPTIONS(EXTERNAL_TABLE=EXECUTE, TRIM=LRTRIM)
LOAD DATA
INFILE 'products'
APPEND
INTO TABLE PRODUCTS
FIELDS TERMINATED BY ","
(
  PRODUCT_ID,
  COUNTRY,
  LABEL,
  DETAILED_LABEL
)
End of generated control file for possible reuse.

created temporary directory object SYS_SQLLDR_XT_TMPDIR_00000 for path /home/oracle/labs
enable parallel DML: ALTER SESSION ENABLE PARALLEL DML
creating external table "SYS_SQLLDR_X_EXT_PRODUCTS"

CREATE TABLE "SYS_SQLLDR_X_EXT_PRODUCTS"
(
  "PRODUCT_ID" NUMBER(6),
  "COUNTRY" CHAR(3),
  "LABEL" VARCHAR2(10),
  "DETAILED_LABEL" VARCHAR2(18)
)
ORGANIZATION external
(
  TYPE oracle_loader
  DEFAULT DIRECTORY SYS_SQLLDR_XT_TMPDIR_00000
  ACCESS PARAMETERS
  (
    RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    BADFILE 'SYS_SQLLDR_XT_TMPDIR_00000':'products_%p.bad'
    LOGFILE 'products_%p.log_xt'
    READSIZE 1048576
    FIELDS TERMINATED BY "," LRTRIM
    REJECT ROWS WITH ALL NULL FIELDS
    (
      "PRODUCT_ID" CHAR(255),
      "COUNTRY" CHAR(255),
      "LABEL" CHAR(255),
      "DETAILED_LABEL" CHAR(255)
    )
  )
location
(
  'products.dat'
)
```

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

```

)
)REJECT LIMIT UNLIMITED
executing INSERT statement to load database table PRODUCTS
INSERT /*+ append parallel(auto) */ INTO PRODUCTS
(
  PRODUCT_ID,
  COUNTRY,
  LABEL,
  DETAILED_LABEL
)
SELECT
  "PRODUCT_ID",
  "COUNTRY",
  "LABEL",
  "DETAILED_LABEL"
FROM "SYS_SQLLDR_X_EXT_PRODUCTS"

dropping external table "SYS_SQLLDR_X_EXT_PRODUCTS"

Table PRODUCTS:
  8 Rows successfully loaded.

Run began on Mon Nov 21 18:39:11 2016
Run ended on Mon Nov 21 18:39:11 2016

Elapsed time was:      00:00:00.55
CPU time was:          00:00:00.02
$
```

13. Question: Which operations did SQL*Loader execute in express mode?
Answer: SQL*Loader first created a temporary external table, used the external table to load the content of the external data file into the table, and finally dropped the temporary external table.
14. In the /home/oracle/labs directory where you are working, find the file named products_nnnn.log_xt that you just created, and display its contents. The date in the file listing will distinguish the right file from the others. For example:

```
$ ls -l products_*.log_xt
-rwxrwxrwx 1 oracle oinstall 698 Dec 20 02:24 products_160140.log_xt
-rw-r--r-- 1 oracle oinstall 698 Jan 9 02:24 products_5026.log_xt

$ cat products_5026.log_xt

LOG file opened at 11/21/16 18:39:11

Field Definitions for table SYS_SQLLDR_X_EXT_PRODUCTS
Record format DELIMITED BY NEWLINE
Data in file has same endianness as the platform
Reject rows with all null fields

Fields in Data Source:

PRODUCT_ID           CHAR ( 255 )
Terminated by ","
Trim whitespace from left and right
COUNTRY              CHAR ( 255 )
Terminated by ","
Trim whitespace from left and right
LABEL                CHAR ( 255 )
Terminated by ","
Trim whitespace from left and right
DETAILED_LABEL       CHAR ( 255 )
Terminated by ","
Trim whitespace from left and right
$
```

Load Data by Using SQL*Loader in Conventional Mode

In this section, you will load data into the SH.INVENTORIES table in PDB2 by using SQL*Loader in conventional mode. Currently, there are 476 rows in the SH.INVENTORIES table.

1. Make sure that your current directory is /home/oracle/labs.

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

2. Start SQL*Loader, connect to PDB2 as the sh user, and load the SH.INVENTORIES table from the \$HOME/labs/DP_inventories.dat data file in conventional mode. See [Appendix - Product-Specific Credentials](#) for the password. The result shows that 83 rows were successfully loaded in the SH.INVENTORIES table.

```
$ sqlldr userid=sh/<password>@PDB2 control=DP_inventories.ctl
log=inventories.log data=DP_inventories.dat

SQL*Loader: Release 12.2.0.1.0 - Production on Mon Nov 21 19:31:05 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Path used:      Conventional
Commit point reached - logical record count 64
Commit point reached - logical record count 83

Table SH.INVENTORIES:
  83 Rows successfully loaded.

Check the log file:
  inventories.log
for more information about the load.
$
```

3. Question: What do you observe about the commit point reached in the results?
Answer: The commit was executed not after every inserted row, but after 64 rows were inserted. This is the default value of the `ROWS` parameter.
4. Start SQL*Plus and connect to PDB2 as the `sh` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus sh/<password>@PDB2

...
SQL>
```

5. Count the number of rows in the `SH.INVENTORIES` table. The result shows 559 rows.

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

 COUNT(*)
 -----
      559
SQL>
```

6. Question: Did SQL*Loader append new rows or replace rows in the `SH.INVENTORIES` table?
Answer: Originally, there were 476 rows in this table. Now there are 559 rows, which means 83 new rows were added, or "appended," by SQL*Loader. The `APPEND` parameter is the default value of the `SQL INSERT` behavior into the table.
7. Exit SQL*Plus.

```
SQL> EXIT

...
$
```

8. View the inventories.log file. Notice that the insert option in effect for the SH.INVENTORIES table is APPEND.

```
$ cat inventories.log

SQL*Loader: Release 12.2.0.1.0 - Production on Mon Nov 21 19:31:05 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Control File: DP_inventories.ctl
Data File: DP_inventories.dat
Bad File: DP_inventories.bad
Discard File: none specified

(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array: 64 rows, maximum of 256000 bytes
Continuation: none specified
Path used: Conventional

Table SH.INVENTORIES, loaded from every logical record.
Insert option in effect for this table: APPEND

      Column Name          Position    Len  Term Encl Datatype
-----+-----+-----+-----+-----+-----+-----+
WAREHOUSE_ID                      FIRST      *   ,     CHARACTER
PRODUCT_ID                         NEXT      *   ,     CHARACTER
QUANTITY_ON_HAND                   NEXT      *   ,     CHARACTER

Table SH.INVENTORIES:
 83 Rows successfully loaded.
 0 Rows not loaded due to data errors.
 0 Rows not loaded because all WHEN clauses were failed.
 0 Rows not loaded because all fields were null.

Space allocated for bind array:           49536 bytes(64 rows)
Read    buffer bytes: 1048576

Total logical records skipped:          0
Total logical records read:            83
Total logical records rejected:        0
Total logical records discarded:       0

Run began on Mon Nov 21 19:31:05 2016
Run ended on Mon Nov 21 19:31:05 2016

Elapsed time was: 00:00:00.39
CPU time was:    00:00:00.02
$
```

9. View the content of the control file named `DP_inventories.ctl` in the vi editor. Notice that the code *appends* data.

```
$ vi DP_inventories.ctl

...
LOAD DATA
infile '/home/oracle/labs/DP_inventories.dat'
INTO TABLE SH.INVENTORIES
APPEND
FIELDS TERMINATED BY ','
(warehouse_id,
product_id,
quantity_on_hand)
$
```

10. Change `APPEND` to `TRUNCATE` so that the control file truncates the table. Save the file and quit the vi editor (`:wq`).

```
...
LOAD DATA
infile '/home/oracle/labs/DP_inventories.dat'
INTO TABLE SH.INVENTORIES
TRUNCATE
FIELDS TERMINATED BY ','
(warehouse_id,
product_id,
quantity_on_hand)
$
```

11. Start SQL*Loader, connect to `PDB2` as the `sh` user, and re-execute the load operation with the `ROWS` parameter set to 10. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlldr userid=sh/<password>@PDB2 control=DP_inventories.ctl
log=inventories.log data=DP_inventories.dat ROWS=10

SQL*Loader: Release 12.2.0.1.0 - Production on Mon Nov 21 19:44:49 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Path used: Conventional
Commit point reached - logical record count 10
Commit point reached - logical record count 20
Commit point reached - logical record count 30
Commit point reached - logical record count 40
Commit point reached - logical record count 50
Commit point reached - logical record count 60
Commit point reached - logical record count 70
Commit point reached - logical record count 80
Commit point reached - logical record count 83

Table SH.INVENTORIES:
 83 Rows successfully loaded.

Check the log file:
  inventories.log
for more information about the load.
$
```

- Start SQL*Plus and connect to PDB2 as the sh user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus sh/<password>@PDB2

...
SQL>
```

- Verify the number of rows in the INVENTORIES table. The table now has 83 rows. The TRUNCATE option had cleared out the original rows in the table and inserted 83 new rows.

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

COUNT(*)
-----
83
SQL>
```

Re-enable the Check Constraint

Suppose a DBA discovers that the CHECK constraint was disabled on the WAREHOUSE_ID column in the SH.INVENTORIES table at the time of the load. This disabled constraint allowed only values within a certain range. Use the \$HOME/labs/DP_check.sql SQL script to empty the table and re-enable the check constraint. Then, reload the table.

- Execute the \$HOME/labs/DP_check.sql script.

```
SQL> @ $HOME/labs/DP_check.sql
```

Connected.

Table truncated.

Table altered.

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

\$

- Start SQL*Loader, connect to PDB2 as the sh user, and reload the table. See [Appendix - Product-Specific Credentials](#) for the password. The results indicate that 20 rows were successfully loaded into the SH.INVENTORIES table.

```
$ sqlldr userid=sh/<password>@PDB2 control=DP_inventories.ctl  
log=inventories.log data=DP_inventories.dat ROWS=10
```

SQL*Loader: Release 12.2.0.1.0 - Production on Thu Nov 24 12:33:26 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Path used: Conventional

Commit point reached - logical record count 10
Commit point reached - logical record count 20
Commit point reached - logical record count 30
Commit point reached - logical record count 40
Commit point reached - logical record count 50
Commit point reached - logical record count 60
Commit point reached - logical record count 70
Commit point reached - logical record count 80

Table SH.INVENTORIES:

20 Rows successfully loaded.

Check the log file:

```
inventories.log  
for more information about the load.  
$
```

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

```
$ cat inventories.log
```

SQL*Loader: Release 12.2.0.1.0 - Production on Thu Nov 24 12:38:49 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Control File: DP_inventories.ctl
Data File: DP_inventories.dat
Bad File: DP_inventories.bad

```

Discard File: none specified
(Allow all discards)

Number to load: ALL
Number to skip: 0
Errors allowed: 50
Bind array:      10 rows, maximum of 256000 bytes
Continuation:    none specified
Path used:       Conventional

Table SH.INVENTORIES, loaded from every logical record.
Insert option in effect for this table: TRUNCATE

```

Column Name	Position	Len	Term	Encl	Datatype
WAREHOUSE_ID	FIRST	*	,		CHARACTER
PRODUCT_ID	NEXT	*	,		CHARACTER
QUANTITY_ON_HAND	NEXT	*	,		CHARACTER

```

Record 21: Rejected - Error on table SH.INVENTORIES.
ORA-02290: check constraint (SH.CK_WAREHOUSE_ID) violated
Record 22: Rejected - Error on table SH.INVENTORIES.
ORA-02290: check constraint (SH.CK_WAREHOUSE_ID) violated
...
Record 71: Rejected - Error on table SH.INVENTORIES.
ORA-02290: check constraint (SH.CK_WAREHOUSE_ID) violated

```

MAXIMUM ERROR COUNT EXCEEDED - Above statistics reflect partial run.

Table SH.INVENTORIES:

```

20 Rows successfully loaded.
51 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
0 Rows not loaded because all fields were null.

```

Space allocated for bind array:	7740 bytes(10 rows)
Read buffer bytes:	1048576

Total logical records skipped:	0
Total logical records read:	80
Total logical records rejected:	51
Total logical records discarded:	0

Run began on Thu Nov 24 12:38:49 2016
 Run ended on Thu Nov 24 12:38:49 2016

Elapsed time was:	00:00:00.13
CPU time was:	00:00:00.03
\$	

4. Question: Did the load try to load all rows?

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

SQL*Loader stopped.

Load Data with SQL*Loader in Direct Mode

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

1. Start SQL*Loader, connect to PDB2 as the sh user, and load the SH.INVENTORIES table in direct mode. See [Appendix - Product-Specific Credentials](#) for the password. The results indicate that the load completed and the record count is 83.

```
$ sqlldr userid=sh/<password>@PDB2 control=DP_inventories.ctl
log=inventories.log data=DP_inventories.dat ROWS=10 DIRECT=TRUE

SQL*Loader: Release 12.2.0.1.0 - Production on Mon Nov 21 19:53:35 2016

Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.

Path used:      Direct
Save data point reached - logical record count 10.
Save data point reached - logical record count 20.
Save data point reached - logical record count 30.
Save data point reached - logical record count 40.
Save data point reached - logical record count 50.
Save data point reached - logical record count 60.
Save data point reached - logical record count 70.
Save data point reached - logical record count 80.

Load completed - logical record count 83.

Table SH.INVENTORIES:
  83 Rows successfully loaded.

Check the log file:
  inventories.log
for more information about the load.
$
```

2. Questions: Does the direct load use the SQL INSERT statement? How does the direct path commit the rows inserted?
Answers: The direct load loads records into the blocks, writing the data blocks directly to the database files. You can observe that there is no COMMIT instruction, but SAVE. During a data save, only full database blocks are written to the database.
3. Question: Did it enforce the CHECK constraint?
Answer: No, it did not. This is the reason why all rows were loaded, regardless of the WAREHOUSE_ID value to be inserted.
4. Question: Does SQL*Loader in direct mode ignore all constraints?
Answer: No, it does not. It enforces PRIMARY KEY, UNIQUE, and NOT NULL constraints.
5. Close the terminal window.

Practice 9-3 Querying External Tables

Overview

In this practice, you query partitioned external tables.

Suppose you received new external files containing records about sales. The sales records are dispatched in two files according to the sales year:

- \$HOME/labs/DP/DP_sales_1998.dat
- \$HOME/labs/DP2/DP2_sales_1999.dat

You don't want to load or insert the records into a table in PDB1, but you want to be able to read the sales data from the external files.

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. If PDB1 does not exist, execute the `$HOME/labs/setup_pdb1.sh` script.

```
$ $HOME/labs/setup_pdb1.sh

...
$
```

3. Execute the `$HOME/labs/DP_glogin.sh` shell script to set formatting for all columns selected in queries and to place both `.dat` files in `DP` and `DP2` subdirectories.

Note: You can ignore the error message about not being able to remove the `orders.dmp` file.

```
$ $HOME/labs/DP_glogin.sh

...
$
```

4. Start SQL*Plus and connect to PDB1 as the `SYSTEM` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB1  
...  
SQL>
```

5. In PDB1, create the SH.SALES_EXT_RANGE external table.

- a. Create two directories in the database that point to where the external files are stored.

```
SQL> CREATE DIRECTORY ext_dir AS '/home/oracle/labs/DP/';  
Directory created.  
  
SQL> CREATE DIRECTORY ext_dir2 AS '/home/oracle/labs/DP2/';  
Directory created.  
SQL>
```

- b. Create an SH schema for the sales data. See [Product-Specific Credentials](#) for the password. Grant the SH user CREATE SESSION and CREATE TABLE privileges. Also grant the SH user READ WRITE privileges on the directories you just created (ext_dir and ext_dir2).

```
SQL> CREATE USER sh IDENTIFIED by <password>;  
User created.  
  
SQL> GRANT create session, create table to sh;  
Grant succeeded.  
  
SQL> GRANT read, write on DIRECTORY ext_dir to sh;  
Grant succeeded.  
  
SQL> GRANT read, write on DIRECTORY ext_dir2 to sh;  
Grant succeeded.  
  
SQL>
```

- c. In case it already exists, drop the SH.SALES_EXT_RANGE table. You should get an error stating that the table does not exist.

```
SQL> DROP TABLE sh.sales_ext_range;  
  
DROP TABLE sh.sales_ext_range  
*  
ERROR at line 1:  
ORA-00942: table or view does not exist  
SQL>
```

- d. Execute the following code to create the structure of the external table SH.SALES_EXT_RANGE. The code partitions the table on the TIME_ID column. You can copy the code from \$HOME/labs/external_table.sql and paste it into SQL*Plus. Or, you run the SQL script by entering @\$HOME/labs/external_table.sql.

```
SQL> CREATE TABLE sh.sales_ext_range
  ( time_id          DATE NOT NULL,
    prod_id           INTEGER NOT NULL,
    cust_id           INTEGER NOT NULL,
    channel_id        INTEGER NOT NULL,
    promo_id          INTEGER NOT NULL,
    quantity_sold     NUMBER(10,2),
    amount_sold       NUMBER(10,2)
  )
ORGANIZATION EXTERNAL
(
  TYPE ORACLE_LOADER
    DEFAULT DIRECTORY ext_dir
    ACCESS PARAMETERS
    (
      RECORDS DELIMITED BY NEWLINE
      BADFILE 'sh%a_%p.bad'
      LOGFILE 'sh%a_%p.log'
      FIELDS TERMINATED BY ','
      MISSING FIELD VALUES ARE NULL
    )
)
PARALLEL
REJECT LIMIT UNLIMITED
PARTITION by range (time_id)
(PARTITION year1998 VALUES LESS THAN (TO_DATE('31-12-1998', 'DD-MM-YYYY'))
LOCATION ('DP_sales_1998.dat'),
PARTITION year1999 VALUES LESS THAN (TO_DATE('31-12-1999', 'DD-MM-YYYY'))
LOCATION (ext_dir2:'DP2_sales_1999.dat'));

...
SQL>
```

- e. Question: Based on the code in the previous step, which directories does the external table use?
 Answer: The partitions of the external table use two directories. The default directory for any partition created is ext_dir. The last partition uses another directory, ext_dir2, which corresponds to the active files for the current sales.
- f. Verify that the locations are correctly set for the partitions by querying the DBA_XTERNAL_LOC_PARTITIONS view.

```
SQL> SELECT table_name, partition_name, location, directory_name
  2  FROM DBA_XTERNAL_LOC_PARTITIONS;

TABLE_NAME      PARTITION_NAME LOCATION          DIRECTORY_NAME
-----          -----
SALES_EXT_RANGE YEAR1998      DP_sales_1998.dat
SALES_EXT_RANGE YEAR1999      DP2_sales_1999.dat EXT_DIR2
SQL>
```

6. Perform counts on the external table.

- a. Count the rows for sales in 1998. The results indicate that 357668 rows were read.

```
SQL> SELECT count(*) FROM sh.sales_ext_range PARTITION (year1998);  
  
COUNT( * )  
-----  
357668  
SQL>
```

- b. Count the rows for sales in 1999. The results indicate that 495890 rows were read.

```
SQL> SELECT count(*) FROM sh.sales_ext_range PARTITION (year1999);  
  
COUNT( * )  
-----  
495890  
SQL>
```

- c. Count the number of rows for sales in both 1998 and 1999. The results indicate that 853558 rows were read.

```
SQL> SELECT count(*) FROM sh.sales_ext_range;  
  
COUNT( * )  
-----  
853558  
SQL>
```

- d. Exit SQL*Plus.

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

7. Issue the following commands to find out whether the number of rows read is equivalent to the number of records that exist in the two external files. The results show that the number of records in the DP_sales_1998.dat file is 357675 and the number of records in the DP2_sales_1999.dat file is 495899. Together, the number of records equals 853574. This value is higher than the number of rows read, which you found to equal 853558 in the previous step.

```
$ wc -l /home/oracle/labs/DP/DP_sales_1998.dat  
357675 /home/oracle/labs/DP/sales_1998.dat  
  
$ wc -l /home/oracle/labs/DP2/DP2_sales_1999.dat  
495899 /home/oracle/labs/DP2/sales_1999.dat  
$
```

8. Check the log files to get the reason for the discrepancy.

- a. List the log files. There are four.

```
$ ls -l /home/oracle/labs/DP/*.log

-rw-r--r-- 1 oracle oinstall 1441 Nov 24 16:32 sh000_11177.log
-rw-r--r-- 1 oracle oinstall 2383 Nov 24 16:32 sh001_11179.log
-rw-r--r-- 1 oracle oinstall 1443 Nov 24 16:32 sh002_11195.log
-rw-r--r-- 1 oracle oinstall 1572 Nov 24 16:32 sh003_11197.log
```

- b. View the content of all log files. According to the log files, there were 16 records that could not be "inserted" into the external table structure because some fields in the external files contained NULL value whereas the column in the table is set to NOT NULL.

```
$ more /home/oracle/labs/DP/*.log

LOG file opened at 11/24/16 16:32:39
...
error processing column TIME_ID in row 50000 for datafile
/home/oracle/labs/DP/DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)error processing column TIME_ID
in row 100000 for datafile /home/oracle/labs/DP/DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)

LOG file opened at 11/24/16 16:32:39
...
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP/DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile /home/oracle/labs/DP/
DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile /home/oracle/labs/DP/
DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile /home/oracle/labs/DP/
DP_sales_1998.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
```

```

...
LOG file opened at 11/24/16 16:32:39
...
error processing column TIME_ID in row 50000 for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)error processing column TIME_ID
in row 100000 for datafile /home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)

LOG file opened at 11/24/16 16:32:39
...
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
error processing column TIME_ID in a row for datafile
/home/oracle/labs/DP2/DP2_sales_1999.dat
ORA-01400: cannot insert NULL into (TIME_ID)
...
$
```

9. Start SQL*Plus and connect to PDB1 as the SYSTEM user. See [Appendix - Product-Specific Credentials](#) for the password.

```

$ sqlplus SYSTEM/<password>@PDB1
...
SQL>
```

10. Attempt to create an index on the partition key of the external table to get better query performance. The resulting error indicates that you cannot create an index on an external organized table.

```

SQL> CREATE INDEX sh.i_ext_sales_time_id ON sh.sales_ext_range (time_id);
CREATE INDEX sh.i_ext_sales_time_id ON sh.sales_ext_range (time_id)
*
ERROR at line 1:
ORA-30657: operation not supported on external organized table
SQL>
```

11. Suppose that a new file with sales for year 2000 has arrived. Add a new partition called year2000 to the table.

```

SQL> ALTER TABLE sh.sales_ext_range ADD PARTITION year2000 VALUES LESS THAN
(TO_DATE('31-12-2000', 'DD-MM-YYYY')) LOCATION (ext_dir2:'DP2_sales_2000.dat');
Table altered.
SQL>
```

12. Count the number of sales rows in the SH.SALES_EXT_RANGE table for year 2000. The number of rows read equals 235893.

```
SQL> SELECT count(*) FROM sh.sales_ext_range PARTITION (year2000);

  COUNT( * )
-----
  235893
SQL>
```

13. Count the actual number of rows in the DP2_sales_2000.dat file. Again, the result indicates that the number of rows read (235893) is less than the number of rows in the data file (236002). The discrepancy may or may not be due to null rows getting discarded, as you observed in previous steps.

```
SQL> host wc -l /home/oracle/labs/DP2/DP2_sales_2000.dat

236002 /home/oracle/labs/DP2/sales_2000.dat
SQL>
```

14. Perform another check on the data. Query the number of rows that have a TIME_ID value that falls within the year 2000. The results show that the database read only one row.

```
SQL> SELECT count(*) FROM sh.sales_ext_range
  2 WHERE time_id <= TO_DATE('31-12-2000','DD-MM-YYYY') AND time_id >=
TO_DATE('01-01-2000', 'DD-MM-YYYY');

  COUNT( * )
-----
    1
SQL>
```

15. Exit SQL*Plus.

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

16. View the contents of the DP2_sales_2000.dat file. Notice that most records do not contain sales for year 2000. Be aware that in Oracle Database 12.2, row validation is not yet supported. You must ensure that the records satisfy the partitioning conditions. If you were to remedy this situation, you would need to create two distinct files: one for 2000 sales and another one for 2001 sales, and then add another partition for 2001 sales.

```
$ cat /home/oracle/labs/DP2/DP2_sales_2000.dat  
24-OCT-01,135,10792,3,999,1,51.43  
24-OCT-01,135,10960,3,999,1,51.43  
24-OCT-01,135,11126,3,999,1,51.43  
24-OCT-01,135,11136,3,999,1,51.43  
24-OCT-01,135,11201,3,999,1,51.43  
24-OCT-01,135,11780,3,999,1,51.43  
24-OCT-01,135,12054,3,999,1,51.43  
24-OCT-01,135,12069,3,999,1,51.43  
24-OCT-01,135,12500,3,999,1,51.43  
24-OCT-01,135,13187,3,999,1,51.43  
24-OCT-01,135,13421,3,999,1,51.43  
24-OCT-01,135,40207,3,999,1,51.43  
...  
$
```

17. Close the terminal window.

Practice 9-4 Unloading External Tables

Overview

In this practice, you will write the OE.ORDERS table data to a dump file using the external tables.

Assumptions

You are logged in to VM1 as the oracle user.

If PDB1 does not exist, execute the \$HOME/labs/setup_pdb1.sh on VM1.

If PDB2 does not exist, execute the \$HOME/labs/setup_pdb2.sh on VM1.

You completed Practice 9-3 Querying External Tables (only steps 1, 2, and 3 are necessary).

Tasks

Complete the following steps.

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 SYSTEM user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus SYSTEM/<password>@PDB1

...
SQL>
```

3. In PDB1, create an external table called OE.ORDERS_EXT that unloads the rows from OE.ORDERS to an external file called orders.dmp. Later, that file will be read from an external table in PDB2.

Note: If you prefer, you can enter @\$HOME/labs/UNLOAD_EXT_TABLES_1.sql to run the SQL script instead of typing the code below.

```
SQL> CREATE TABLE oe.orders_ext
  2  ORGANIZATION EXTERNAL
  3  (TYPE ORACLE_DATAPUMP
  4  DEFAULT DIRECTORY ext_dir
  5  LOCATION ('orders.dmp'))
  6  AS SELECT * FROM oe.orders;

Table created.

SQL>
```

4. Verify that the external file (`orders.dmp`) is listed in the `/home/oracle/labs/DP` directory. The result indicates that it is listed. Your date will be different than the one shown below.

```
SQL> HOST ls -l /home/oracle/labs/DP/orders.dmp
-rw-r----- 1 oracle oinstall 16384 Nov 24 17:27 /home/oracle/labs/DP/orders.dmp
SQL>
```

5. Count the number of rows in the `OE.ORDERS_EXT` table. The number of rows read is 105.

```
SQL> SELECT count(*) FROM oe.orders_ext;
          COUNT(*)
-----
          105
SQL>
```

6. Connect to PDB2 as the `SYSTEM` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CONNECT SYSTEM/<password>@PDB2
Connected.
SQL>
```

7. Create a user named `oe`. See [Appendix - Product-Specific Credentials](#) for the password.

```
SQL> CREATE USER oe IDENTIFIED BY <password>;
User created.
SQL>
```

8. Create an external directory named `ext_dir`.

```
SQL> CREATE DIRECTORY ext_dir AS '/home/oracle/labs/DP/';
Directory created.
SQL>
```

9. Create an external table named `OE.ORDERS_EXT` that loads `orders.dmp`.

Note: If you prefer, you can enter `@$HOME/labs/UNLOAD_EXT_TABLES_2.sql` to run the SQL script instead of typing the code below.

```
SQL> CREATE TABLE oe.orders_ext
  2  (order_id NUMBER, order_date TIMESTAMP(6) WITH LOCAL TIME ZONE,
  3  order_mode VARCHAR2(8), customer_id NUMBER(6),
  4  order_status NUMBER(2), order_total NUMBER(8),
  5  sales_rep_id NUMBER(6), promotion_id NUMBER(6))
  6  ORGANIZATION EXTERNAL
  7  (TYPE ORACLE_LOADER DEFAULT DIRECTORY ext_dir
  8  LOCATION ('orders.dmp'));
```

Table created.

SQL>

10. Try to query the entire OE.ORDERS_EXT table. You get an error.

```
SQL> SELECT * FROM oe.orders_ext;

select * from oe.orders_ext
*
ERROR at line 1:
ORA-29913: error in executing ODCIEXTTABLEFETCH callout
ORA-30653: reject limit reached
SQL>
```

11. Question: Which type of access driver was used to unload the data from the table into an external file?
Answer: The access driver was ORACLE_DATAPUMP. The binary file (orders.dmp) created has the same format as the files used by the Data Pump Import and Export utilities and can be interchanged with them. During the loading (reading from the external table), the same access driver must be used.
12. Recreate the external table with the appropriate access driver.

- a. Drop the OE.ORDERS_EXT table that you just created.

```
SQL> DROP TABLE oe.orders_ext;

Table dropped.
SQL>
```

- b. Create the OE.ORDERS_EXT table again, and this time, specify TYPE ORACLE_DATAPUMP (see line 7 below) instead of what you used before, which was TYPE ORACLE_LOADER.
Note: If you prefer, you can enter @\$HOME/labs/UNLOAD_EXT_TABLES_3.sql to run the SQL script instead of typing the code below.

```
SQL> CREATE TABLE oe.orders_ext
  2  (order_id NUMBER, order_date TIMESTAMP(6) WITH LOCAL TIME ZONE,
  3   order_mode VARCHAR2(8), customer_id NUMBER(6),
  4   order_status NUMBER(2), order_total NUMBER(8),
  5   sales_rep_id NUMBER(6), promotion_id NUMBER(6))
  6   ORGANIZATION EXTERNAL
  7   (TYPE ORACLE_DATAPUMP DEFAULT DIRECTORY ext_dir LOCATION
  ('orders.dmp'));
```

Table created.

SQL>

13. Query the entire OE.ORDERS_EXT table again. This time, the query returns 105 rows.

```
SQL> SELECT * FROM oe.orders_ext;
```

ORDER_ID	ORDER_DATE	ORDER_MODE	CUSTOMER_ID	ORDER_STATUS	ORDER_TOTAL	SALES REP_ID
5458	16-AUG-07 02.34.12.234359 PM	direct	101	0	78280	153
5457	31-OCT-07 10.22.16.162632 PM	direct	118	5	21586	159

105 rows selected.

SQL>

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

```
SQL> EXIT
```

...

\$

10

Backup and Recovery Concepts



ORACLE®

Objectives for Lesson 10

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

After completing this lesson, you should be able to:

- Identify DBA responsibilities regarding database backup and recovery
- Identify the types of failure that can occur in an Oracle database
- Describe instance recovery
- Describe complete and incomplete recovery



DBA Responsibilities

- Protect the database from failure wherever possible
- Increase the mean time between failures (MTBF)
- Protect critical components by using redundancy
- Decrease the mean time to recover (MTTR)
- Minimize the loss of data



The database administrator (DBA) is typically responsible for ensuring that the database is open and available when users need it. To achieve that goal, the DBA (working with the system administrator):

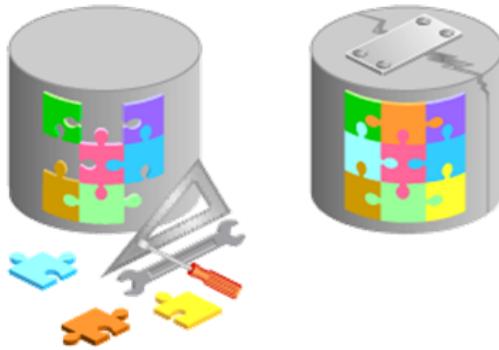
- Anticipates and works to protect the database from failure wherever possible, and to avoid common causes of failure
- Works to increase the mean time between failures (MTBF) that negatively affect availability
- Ensures that hardware is as reliable as possible, that critical components are protected by redundancy, and that operating system maintenance is performed in a timely manner. Oracle Database provides advanced configuration options to increase MTBF, including:
 - Real Application Clusters
 - Oracle Data Guard
- Decreases the mean time to recover (MTTR) by practicing recovery procedures in advance and configuring backups so that they are readily available when needed
- Minimizes the loss of data. DBAs who follow accepted best practices can configure their databases so that no committed transaction is ever lost. Entities that assist in guaranteeing this include:
 - Archive log files (discussed later in this lesson)
 - Flashback technology
 - Standby databases and Oracle Data Guard (discussed in the *Oracle Database 12c: Data Guard Administration* course)

Categories of Failure

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Failures can generally be divided into the following categories:

- Statement failure
- User process failure
- Network failure
- User error
- Instance failure
- Media failure



ORACLE®

Failures can generally be divided into the following categories:

- **Statement failure** : A single database operation (select, insert, update, or delete) fails.
- **User process failure** : A single database session fails.
- **Network failure** : Connectivity to the database is lost.
- **User error** : A user successfully completes an operation, but the operation (dropping a table or entering bad data) is incorrect.
- **Instance failure** : The database instance shuts down unexpectedly.
- **Media failure** : A loss of any file that is needed for database operation (that is, the files have been deleted or the disk has failed).

Statement Failure

Typical Problems	Possible Solutions
Attempts to enter invalid data into a table	Work with users to validate and correct data.
Attempts to perform operations with insufficient privileges	Provide appropriate object or system privileges.
Attempts to allocate space that fails.	<ul style="list-style-type: none">Enable resumable space allocation.Increase owner quota.Add space to tablespace.
Logic errors in applications	Work with developers to correct program errors.



When a single database operation fails, DBA involvement may be necessary to correct errors with user privileges or database space allocation. DBAs may also need to assist in troubleshooting, even for problems that are not directly in their task area. This can vary greatly from one organization to another.

For example, in organizations that use off-the-shelf applications (that is, organizations that have no software developers), the DBA is the only point of contact and must examine logic errors in applications.

To understand logic errors in applications, you should work with developers to understand the scope of the problem. Oracle Database tools may provide assistance by helping to examine audit trails or previous transactions.

Note: In many cases, statement failures are by design and desired. For example, security policies and quota rules are often decided upon in advance. If a user gets an error while trying to exceed his or her limits, it may be desired for the operation to fail and no resolution may be necessary.

User Process Failure

Typical Problems	Possible Solutions
A user performs an abnormal disconnect.	A DBA's action is not usually needed to resolve user process failures.
A user's session is abnormally terminated.	<ul style="list-style-type: none">• Instance background processes roll back uncommitted changes and release locks.
A user experiences a program error that terminates the session.	<ul style="list-style-type: none">• Watch for trends.



User processes that abnormally disconnect from the instance may have uncommitted work in progress that needs to be rolled back. The Process Monitor (PMON) background process periodically polls server processes to ensure that their sessions are still connected. If PMON finds a server process whose user is no longer connected, PMON recovers from any ongoing transactions; it also rolls back uncommitted changes and releases any locks that are held by the failed session.

A DBA's intervention should not be required to recover from user process failure, but the administrator must watch for trends. One or two users disconnecting abnormally is not a cause for concern. A small percentage of user process failures may occur from time to time.

But consistent and systemic failures indicate other problems. A large percentage of abnormal disconnects may indicate a need for user training (which includes teaching users to log out rather than just terminate their programs). It may also be indicative of network or application problems.

Network Failure

Typical Problems	Possible Solutions
Listener fails	Configure a backup listener and connect-time failover
Network Interface Card (NIC) fails.	Configure multiple network cards.
Network connection fails.	Configure a backup network connection.



The best solution to network failure is to provide redundant paths for network connections. Backup listeners, network connections, and network interface cards reduce the chance that network failures will affect system availability.

User Error

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Typical Causes	Possible Solutions
User inadvertently deletes or modifies data.	Roll back transaction and dependent transactions or rewind table.
User drops a table.	Recover table from recycle bin. Recover table from a backup.

- Use Oracle LogMiner to query your online redo logs and archived redo logs through an Enterprise Manager or SQL interface.



ORACLE®

Users may inadvertently delete or modify data. If they have not yet committed or exited their program, they can simply roll back.

You can use Oracle LogMiner to query your online redo logs and archived redo logs through an Enterprise Manager or SQL interface. Transaction data may persist in online redo logs longer than it persists in undo segments; if you have configured archiving of redo information, redo persists until you delete the archived files. Oracle LogMiner is discussed in the *Oracle Database Utilities* reference.

Users who drop a table can recover it from the recycle bin by flashing back the table to before the drop.

If the recycle bin has already been purged, or if the user dropped the table with the PURGE option, the dropped table can still be recovered by using point-in-time recovery (PITR) if the database has been properly configured.

In Oracle Database 12c, RMAN enables you to recover one or more tables or table partitions to a specified point in time without affecting the remaining database objects.

Note: Flashback technologies, PITR, and table recovery are discussed in the Oracle Database 12c: Backup and Recovery Workshop course and in the *Oracle Database Backup and Recovery User's Guide*.

Instance Failure

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASS ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Typical Causes	Possible Solutions
Power outage	Restart the instance by using the <code>STARTUP</code> command. Recovering from instance failure is automatic, including rolling forward changes in the redo logs and then rolling back any uncommitted transactions.
Hardware failure	
Failure of one of the critical background processes	Investigate the causes of failure by using the alert log, trace files, and Enterprise Manager
Emergency shutdown procedures	



Instance failure occurs when the database instance is shut down before synchronizing all database files. An instance failure can occur because of hardware or software failure or through the use of the emergency `SHUTDOWN ABORT` and `STARTUP FORCE` shutdown commands.

Administrator involvement in recovering from instance failure is rarely required if Oracle Restart is enabled and is monitoring your database. Oracle Restart attempts to restart your database instance as soon as it fails. If manual intervention is required, then there may be a more serious problem that prevents the instance from restarting, such as a memory CPU failure.

Media Failure

Typical Causes	Possible Solutions
Failure of disk drive	1. Restore the affected file from backup. 2. Inform the database about a new file location (if necessary). 3. Recover the file by applying redo information (if necessary).
Failure of disk controller	
Deletion or corruption of a file needed for database operation	



Oracle Corporation defines media failure as any failure that results in the loss or corruption of one or more database files (data, control, or redo log file).

Recovering from media failure requires that you restore and recover the missing files.

Understanding Instance Recovery

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

You can understand instance recovery by becoming familiar with these concepts and procedures:

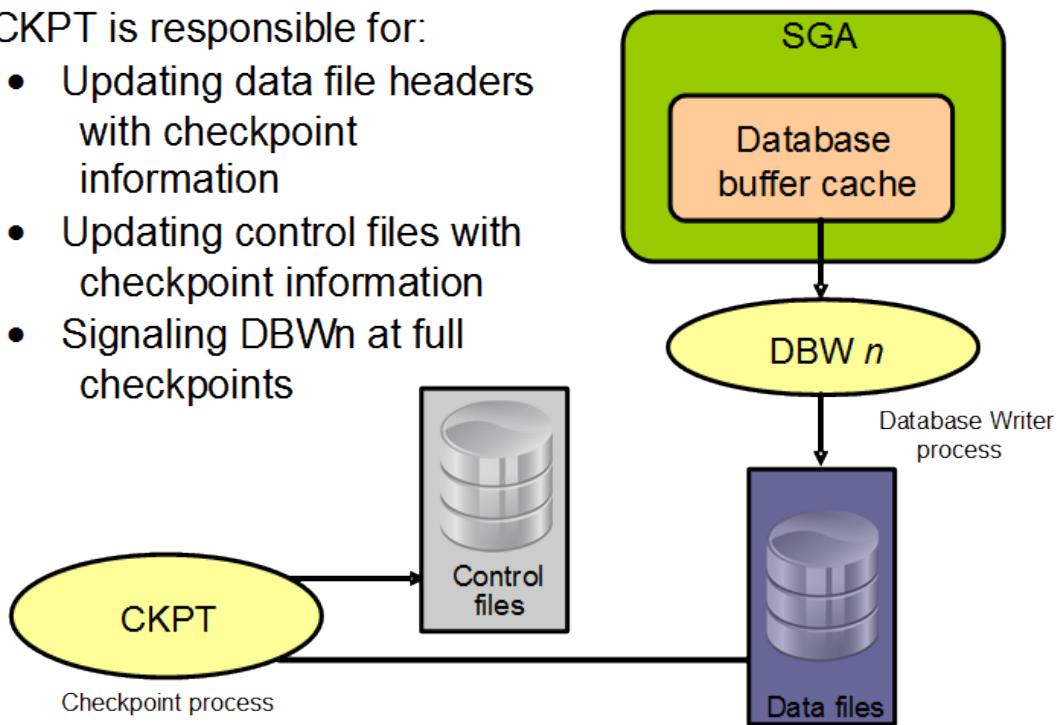
- The Checkpoint Process
- Redo Log Files and Log Writer
- Automatic Instance or Crash Recovery
- Phases of Instance Recovery
- Tuning Instance Recovery
- Using the MTTR Advisor



The Checkpoint Process

CKPT is responsible for:

- Updating data file headers with checkpoint information
- Updating control files with checkpoint information
- Signaling DBWn at full checkpoints



ORACLE®

To understand instance recovery, you need to understand the functioning of certain background processes.

Every three seconds (or more frequently), the CKPT process stores data in a control file to document the modified data blocks that DBWn has written from the SGA to disk. This is called an “incremental checkpoint.” The purpose of a checkpoint is to identify that place in the online redo log file where instance recovery is to begin (which is called the “checkpoint position”).

In the event of a log switch, the CKPT process also writes this checkpoint information to the headers of data files.

Checkpoints exist for the following reasons:

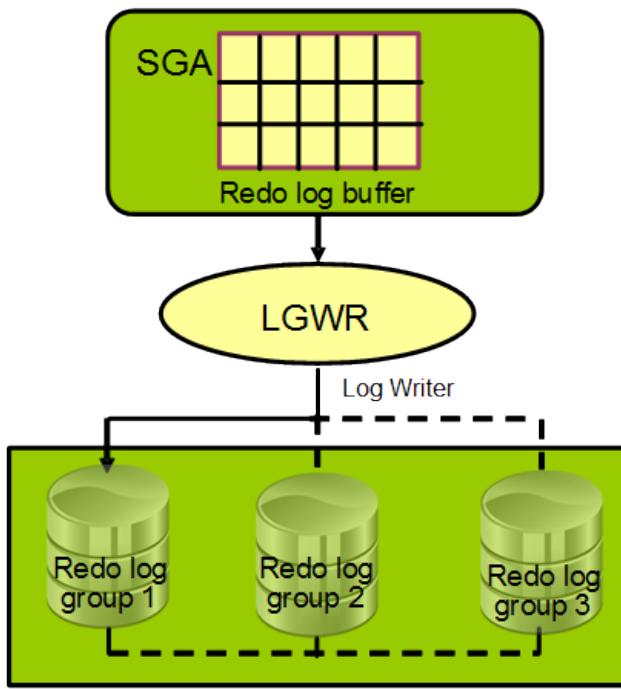
- To ensure that modified data blocks in memory are written to the disk regularly so that data is not lost in case of a system or database failure
- To reduce the time required for instance recovery (Only the online redo log file entries following the last checkpoint need to be processed for recovery.)
- To ensure that all committed data has been written to data files during shutdown

The checkpoint information written by the CKPT process includes checkpoint position, system change number (SCN), location in the online redo log file to begin recovery, information about logs, and so on.

Note: The CKPT process does not write data blocks to the disk or redo blocks to the online redo log files.

Redo Log Files and Log Writer

THESE eKIT MATERIALS ARE FOR YOUR USE IN CLASS ONLY. COPYING OR DISTRIBUTION FROM THIS COMPUTER IS STRICTLY PROHIBITED.



Redo log files:

- Record changes to the database
- Should be multiplexed to protect against loss

Log Writer (LGWR) writes:

- At commit
- When one-third full
- Every three seconds
- Before DBW n writes
- Before clean shutdowns

ORACLE®

Redo log files record changes to the database as a result of transactions and internal Oracle server actions. (A transaction is a logical unit of work consisting of one or more SQL statements run by a user.) Redo log files protect the database from loss of integrity because of system failures caused by power outages, disk failures, and so on. Redo log files should be multiplexed to ensure that the information stored in them is not lost in the event of a disk failure.

The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group, and each group is identified by a number. The Log Writer (LGWR) process writes redo records from the redo log buffer to all members of a redo log group until the files are filled or a log switch operation is requested.

It then switches and writes to the files in the next group. Redo log groups are used in a circular fashion.

Best practice tip: If possible, multiplexed redo log files should reside on different disks.

Automatic Instance or Crash Recovery

Automatic instance or crash recovery:

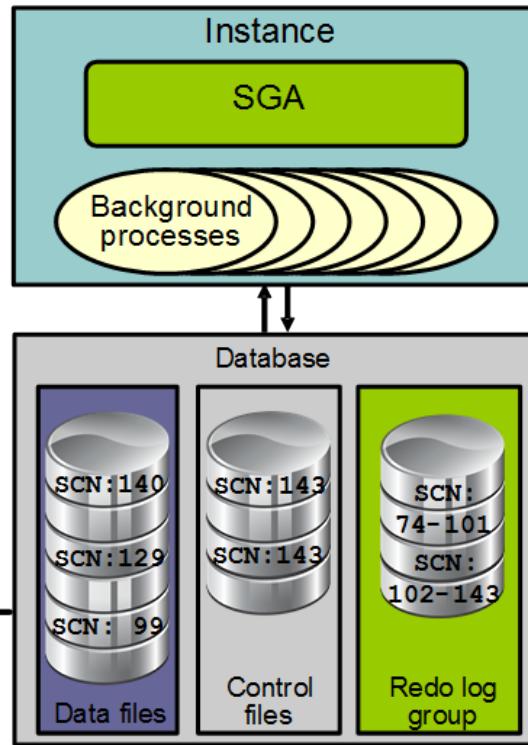
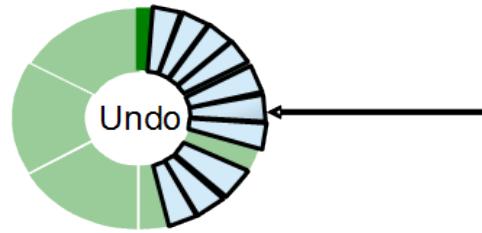
- Is caused by attempts to open a database whose files are not synchronized on shutdown
- Uses information stored in redo log groups to synchronize files
- Involves two distinct operations:
 - Rolling forward: Redo log changes (both committed and uncommitted) are applied to data files.
 - Rolling back: Changes that are made but not committed are returned to their original state.



The Oracle database automatically recovers from instance failure. All that needs to happen is for the instance to be started normally. If Oracle Restart is enabled and configured to monitor this database, then this happens automatically. The instance mounts the control files and then attempts to open the data files. When it discovers that the data files have not been synchronized during shutdown, the instance uses information contained in the redo log groups to roll the data files forward to the time of shutdown. Then the database is opened and any uncommitted transactions are rolled back.

Phases of Instance Recovery

1. Instance startup (data files are out of sync)
2. Roll forward (redo)
3. Committed and uncommitted data in files
4. Database opened
5. Roll back (undo)
6. Committed data in files



ORACLE®

For an instance to open a data file, the system change number (SCN) contained in the data file's header must match the current SCN that is stored in the database's control files.

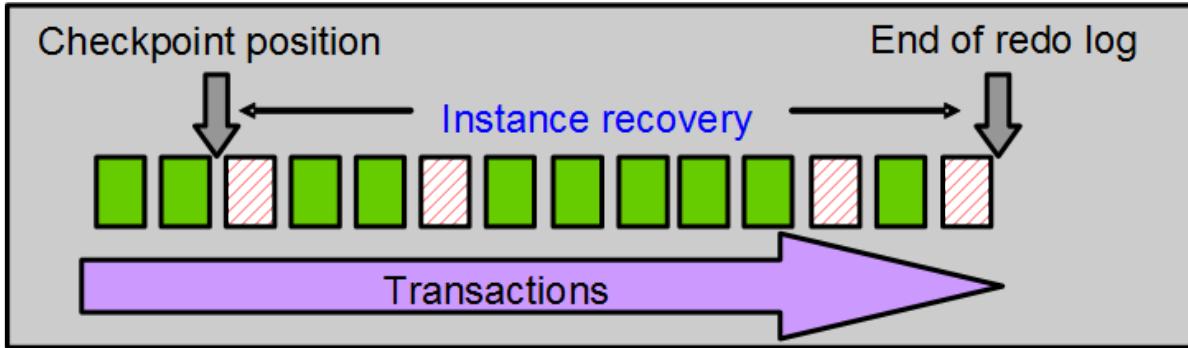
If the numbers do not match, the instance applies redo data from the online redo logs, sequentially "redoing" transactions until the data files are up-to-date. After all data files have been synchronized with the control files, the database is opened and users can log in.

When redo logs are applied, all transactions are applied to bring the database up to the state as of the time of failure. This usually includes transactions that are in progress but have not yet been committed. After the database has been opened, those uncommitted transactions are rolled back.

At the end of the rollback phase of instance recovery, the data files contain only committed data.

Tuning Instance Recovery

- During instance recovery, the transactions between the checkpoint position and the end of redo log must be applied to data files.
- You tune instance recovery by controlling the difference between the checkpoint position and the end of redo log.



ORACLE®

Transaction information is recorded in the redo log groups before the instance returns commit complete for a transaction. The information in the redo log groups guarantees that the transaction can be recovered in case of a failure. The transaction information must also be written to the data file. The data file write usually happens at some time after the information is recorded in redo log groups because the data file write process is much slower than the redo writes. (Random writes for data files are slower than serial writes for redo log files.)

Every three seconds, the checkpoint process records information in the control file about the checkpoint position in the redo log. Therefore, the Oracle Database server knows that all redo log entries recorded before this point are not necessary for database recovery. In the graphic in the slide, the striped blocks have not yet been written to the disk.

The time required for instance recovery is the time required to bring data files from their last checkpoint to the latest SCN recorded in the control file. The administrator controls that time by setting an MTTR target (in seconds) and through the sizing of redo log groups. For example, for two redo groups, the distance between the checkpoint position and the end of the redo log group cannot be more than 90% of the smallest redo log group.

Using the MTTR Advisor

- Specify the desired time in seconds or minutes.
- The default value is 0 (disabled).
- The maximum value is 3,600 seconds (one hour).

The screenshot shows the 'Recovery Settings' page in Oracle Database. At the top right, it says 'Logged in as DBA1'. Below that are three buttons: 'Show SQL', 'Revert', and 'Apply'. The main area is titled 'Instance Recovery'. It contains a note about fast-start checkpointing and the FAST_START_MTTR_TARGET parameter. Below the note, there's a section for 'Current Estimated Mean Time To Recover (seconds)' which shows '19'. Underneath is a 'Desired Mean Time To Recover' input field with '0' entered, followed by a dropdown menu set to 'Minutes'.

ORACLE®

THESE eKIT MATERIALS ARE FOR YOUR USE ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

The FAST_START_MTTR_TARGET initialization parameter simplifies the configuration of recovery time from instance or system failure. The MTTR Advisor converts the FAST_START_MTTR_TARGET value into several parameters to enable instance recovery in the desired time (or as close to it as possible). Please note, explicitly setting the FAST_START_MTTR_TARGET parameter to 0 disables the MTTR Advisor.

The FAST_START_MTTR_TARGET parameter must be set to a value that supports the service level agreement for your system. A small value for the MTTR target increases I/O overhead because of additional data file writes (affecting the performance). However, if you set the MTTR target too large, the instance takes longer to recover after a crash.

For assistance in setting the MTTR target by using Enterprise Manager Cloud Control, navigate as follows:

- Performance > Advisors Home > MTTR Advisor
- Availability > Backup & Recovery > Recovery Settings

Understanding Types of Backups

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

You can understand different types of backups by becoming familiar with these concepts:

- Backup Terminology
- Types of Backups
- RMAN Backup Types



Backup Terminology

- *Backup strategy* may include:
 - Entire database (whole)
 - Portion of the database (partial)
- *Backup type* may indicate inclusion of:
 - All data blocks within your chosen files (full)
 - Only information that has changed since a previous backup (incremental)
 - Cumulative (changes since last level 0)
 - Differential (changes since last incremental)
- *Backup mode* may be:
 - Offline (consistent, cold)
 - Online (inconsistent, hot)



ORACLE®

Whole database backup: Includes all data files and at least one control file (Remember that all control files in a database are identical.)

Partial database backup: May include zero or more tablespaces and zero or more data files; may or may not include a control file

Full backup: Makes a copy of each data block that contains data and that is within the files being backed up

Incremental backup: Makes a copy of all data blocks that have changed since a previous backup. Oracle Database supports two levels of incremental backup (0 and 1). A level 1 incremental backup can be one of two types: cumulative or differential. A cumulative backup backs up all changes since the last level 0 backup. A differential backup backs up all changes since the last incremental backup (which could be either a level 0 or level 1 backup). Change Tracking with RMAN supports incremental backups.

Offline backups: Are also known as “cold” or consistent backup, and are taken while the database is not open. They are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

Online backups: Are also known as “hot” or inconsistent backup, and are taken while the database is open. They are inconsistent because, with the database open, there is no guarantee that the data files are synchronized with the control files.

Types of Backups

Backups may be stored as:

- Image copies
- Backup sets

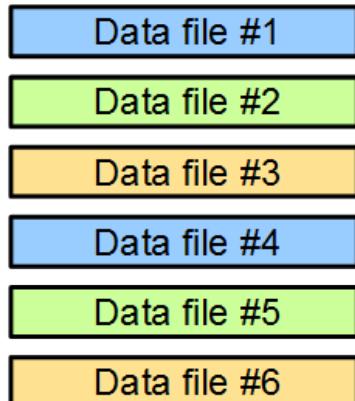
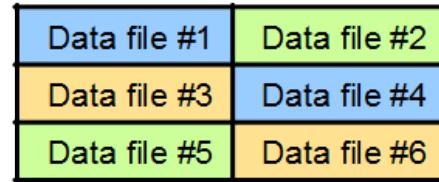


Image copies

(Duplicate data and log files in OS format)



Backup set

(Binary, compressed files in Oracle proprietary format)

ORACLE®

Image copies: Are duplicates of data or archived log files (similar to simply copying the files by using operating system commands)

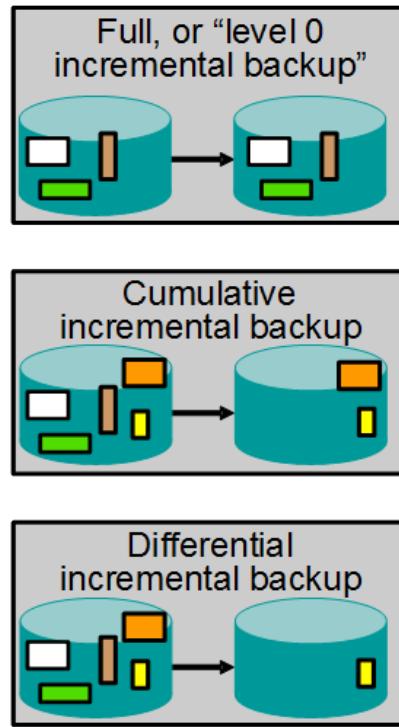
Backup sets: Are collections of one or more binary files that contain one or more data files, control files, server parameter files, or archived log files. With backup sets, empty data blocks are not stored, thereby causing backup sets to use less space on the disk or tape. Backup sets can be compressed to further reduce the space requirements of the backup.

Image copies must be backed up to the disk. Backup sets can be sent to the disk or directly to the tape.

The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy, only the file or files need to be retrieved from your backup location. With backup sets, the entire backup set must be retrieved from your backup location before you extract the file or files that are needed.

The advantage of creating backups as backup sets is better space usage. In most databases, 20% or more of the data blocks are empty blocks. Image copies back up every data block, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems, the advantages of backup sets outweigh the advantages of image copies.

RMAN Backup Types



ORACLE®

Full Backups

A full backup is different from a whole database backup. A full data file backup is a backup that includes every used data block in the file. RMAN copies all blocks into the backup set or image copy, skipping only those data file blocks that are not part of an existing segment. For a full image copy, the entire file contents are reproduced exactly. A full backup cannot be part of an incremental backup strategy; it cannot be the parent for a subsequent incremental backup.

Incremental Backups

An incremental backup is either a level 0 backup, which includes every block in the data files except blocks that have never been used, or a level 1 backup, which includes only those blocks that have been changed since a previous backup was taken. A level 0 incremental backup is physically identical to a full backup. The only difference is that the level 0 backup (as well as an image copy) can be used as the base for a level 1 backup, but a full backup can never be used as the base for a level 1 backup.

Incremental backups are specified using the `INCREMENTAL` keyword of the `BACKUP` command. You specify:

```
INCREMENTAL LEVEL [0 | 1]
```

RMAN can create multilevel incremental backups as follows:

- **Differential:** Is the default type of incremental backup that backs up all blocks changed after the most recent

incremental backup at either level 1 or level 0

- **Cumulative:** Backs up all blocks changed after the most recent backup at level 0

Examples

To perform an incremental backup at level 0, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

To perform a differential incremental backup, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 DATABASE;
```

To perform a cumulative incremental backup, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
```

RMAN makes full backups by default if neither FULL nor INCREMENTAL is specified. Unused block compression causes never-written blocks to be skipped when backing up data files to backup sets, even for full backups.

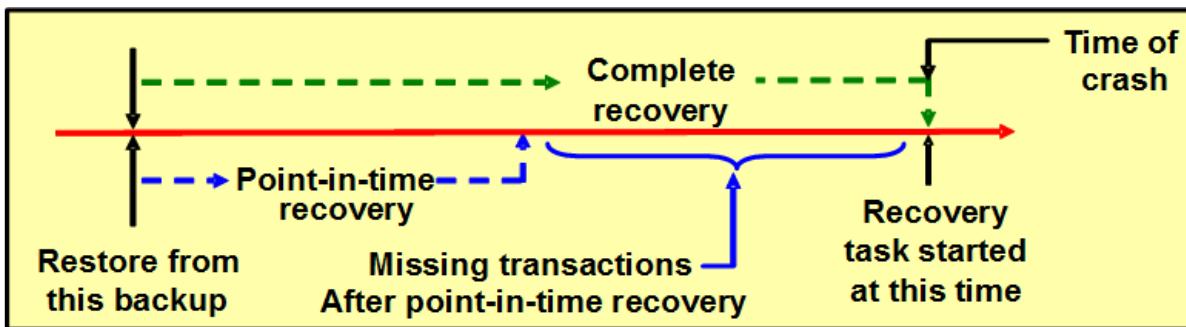
A full backup has no effect on subsequent incremental backups, and is not considered part of any incremental backup strategy, although a full image copy backup can be incrementally updated by applying incremental backups with the RECOVER command.

Note: It is possible to perform any type of backup (full or incremental) of a database that is in NOARCHIVELOG mode—if, of course, the database is not open. Note also that recovery is limited to the time of the last backup. The database can be recovered to the last committed transaction only when the database is in ARCHIVELOG mode.

Comparing Complete and Incomplete Recovery

Recovery can have two kinds of scope:

- **Complete recovery:** Brings the database or tablespace up to the present, including all committed data changes made to the point in time when the recovery was requested
- **Incomplete or point-in-time recovery (PITR):** Brings the database or tablespace up to a specified point in time in the past, before the recovery operation was requested

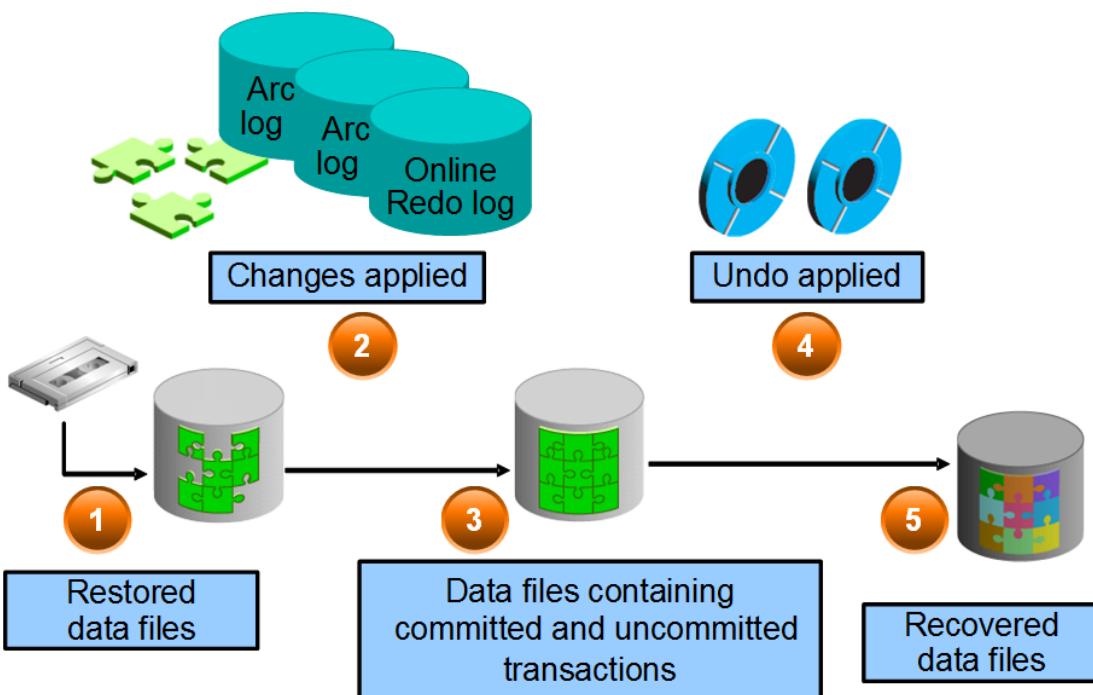


ORACLE®

When you perform complete recovery, you bring the database to the state where it is fully up-to-date, including all committed data modifications to the present time.

Incomplete recovery, however, brings the database or tablespace to some point of time in the past. This is also known as “point-in-time recovery (PITR).” It means there are missing transactions; any data modifications done between the recovery destination time and the present are lost. In many cases, this is the desirable goal because there may have been some changes made to the database that need to be undone. Recovering to a point in the past is a way to remove the unwanted changes.

Complete Recovery Process

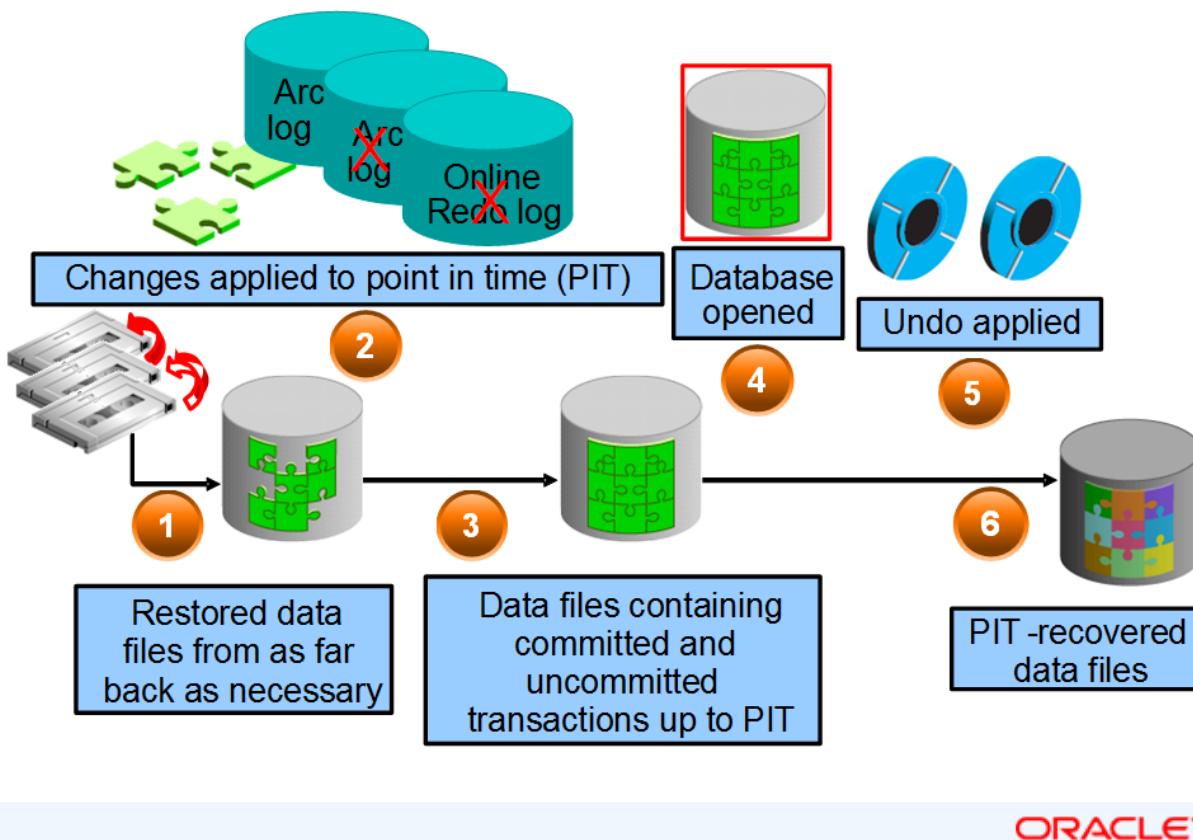


ORACLE®

The following steps describe what takes place during complete recovery:

1. Damaged or missing files are restored from a backup.
2. Changes from incremental backups, archived redo log files, and online redo log files are applied as necessary. The redo log changes are applied to the data files until the current online log is reached and the most recent transactions have been re-entered. Undo blocks are generated during this entire process. This is referred to as rolling forward or cache recovery.
3. The restored data files may now contain committed and uncommitted changes.
4. The undo blocks are used to roll back any uncommitted changes. This is sometimes referred to as transaction recovery.
5. The data files are now in a recovered state and are consistent with the other data files in the database.

Point-in-Time Recovery Process



Incomplete recovery, or database point-in-time recovery (DBPITR), uses a backup to produce a noncurrent version of the database. That is, you do not apply all of the redo records generated after the most recent backup. Perform this type of recovery only when absolutely necessary. To perform point-in-time recovery, you need:

- A valid offline or online backup of all the data files made before the recovery point
- All archived logs from the time of the backup until the specified time of recovery

The steps to perform a point-in-time recovery are as follows:

1. **Restore the data files from backup:** The backup that is used must be from before your target recovery point. This entails either copying files using OS commands or using the `RMAN RESTORE` command.
2. **Use the RECOVER command:** Apply redo from the archived redo log files, including as many as necessary to reach the restore point destination.
3. **State of over-recovery:** Now the data files contain some committed and some uncommitted transactions because the redo can contain uncommitted data.
4. **Use the ALTER DATABASE OPEN command:** The database is opened before undo is applied. This is to provide higher availability.
5. **Apply undo data:** While the redo was being applied, redo supporting the undo data files was also applied. So the undo is available to be applied to the data files in order to undo any uncommitted transactions. That is done next.
6. **Process complete:** The data files are now recovered to the point in time that you chose.

Oracle Flashback Database is the most efficient alternative to DBPITR. Unlike the other flashback features, it operates at a physical level and reverts the current data files to their contents at a past time. The result is like the result of a DBPITR, including the OPEN RESETLOGS, but Flashback Database is typically faster because it does not require you to restore data files and requires only limited application of redo compared to media recovery.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle Data Protection Solutions

Backup and Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Hours/Days	Recovery Manager Oracle Secure Backup
Logical data protection	Minutes/Hours	Flashback Technologies
Recovery analysis	Minimize time for problem identification and recovery planning	Data Recovery Advisor

Disaster Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Seconds/Minutes	Data Guard Active Data Guard



Oracle provides an appropriate data protection solution depending on your backup and recovery objective and RTO:

- Oracle Recovery Manager (RMAN) is the core Oracle Database software component that manages database backup, restore, and recovery processes.
- Oracle Secure Backup (OSB) is Oracle's enterprise-grade tape backup management solution for both database and file system data.
- Oracle Database Flashback technologies are a set of data recovery solutions that enable human errors to be reversed by selectively and efficiently undoing the effects of a mistake.
- The Data Recovery Advisor provides intelligent database problem identification and recovery capabilities.
- Data Guard and Active Data Guard enable physical standby databases to be open for read access while being kept synchronized with the production database through media recovery.

Flashback Technology

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Use Flashback technology for:

- Viewing past states of data
- Winding data back and forth in time
- Assisting users in error analysis and recovery

For error analysis:	For error recovery:
Oracle Flashback Query	Oracle Flashback Transaction Backout
Oracle Flashback Versions Query	Oracle Flashback Table
Oracle Flashback Transaction Query	Oracle Flashback Drop
	Oracle Flashback Database



Oracle Database includes Oracle Flashback technology: a group of features that support viewing past states of data—and winding data back and forth in time—without requiring restoring the database from backup. With this technology, you help users analyze and recover from errors. For users who have committed erroneous changes, use the following to analyze the errors:

- **Flashback Query:** View committed data as it existed at some point in the past. The `SELECT` command with the `AS OF` clause references a time in the past through a time stamp or system change number (SCN).
- **Flashback Version Query:** View committed historical data for a specific time interval. Use the `VERSIONS BETWEEN` clause of the `SELECT` command (for performance reasons with existing indexes).
- **Flashback Transaction Query:** View all database changes made at the transaction level.

Possible solutions to recover from user error:

- **Flashback Transaction Backout:** Rolls back a specific transaction and dependent transactions
- **Flashback Table:** Rewinds one or more tables to their contents at a previous time without affecting other database objects
- **Flashback Drop:** Reverses the effects of dropping a table by returning the dropped table from the recycle bin to the database along with dependent objects such as indexes and triggers
- **Flashback Database:** Returns the database to a past time or SCN

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

Summary for Lesson 10

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

After completing this lesson, you should be able to:

- Identify DBA responsibilities regarding database backup and recovery
- Identify the types of failure that can occur in an Oracle database
- Describe instance recovery
- Describe complete and incomplete recovery



ORACLE®

Practice 10 Overview

- 10-1: Configuring Your Database for Recovery
- 10-2: Backing up the Control File
- 10-3: Configuring Automatic Backups of the Control File and SPFILE
- 10-4: Creating a Whole Database Backup
- 10-5: Creating a Partial Database Backup
- 10-6: Recovering from an essential data file loss
- 10-7: Recovering from an application data file loss



ORACLE®

Practice 10-1 Configuring Your Database for Recovery

Overview

In this practice, you learn how to configure your database to make recovery cases easier. You configure another control file, the Fast Recovery Area (FRA), redo log groups, ARCHIVELOG mode, and redundant archive log destinations.

Tip:

A *control file* is a small binary file that describes the structure of the database. It must be available for writing by the Oracle server whenever the database is mounted or opened. Without this file, the database cannot be mounted, and recovery or re-creation of the control file is required. Your database should have a minimum of two control files on different storage devices to minimize the impact of a loss of one control file. The loss of a single control file causes the instance to fail because all control files must be available at all times. However, recovery can be a simple matter of copying one of the other control files. The loss of all control files is slightly more difficult to recover from, but is not usually catastrophic.

How to configure your database for recovery:

- Ensure redundancy of control files. In case a control file is damaged or lost, recovery is easy when you still have one left.
- Review the fast recovery area configuration.
- Ensure that there are at least two redo log members in each group. In case a redo log member is damaged or lost in a group, recovery is easy when you still have one member left in a group. It can be transparent for end users.
- Place your database in ARCHIVELOG mode. In all cases, you will be able to recover the database either completely or incompletely depending on which database files have been damaged or lost.
- Configure redundant archive log destinations. In cases where you lost archive log files and you need them to recover the database, you will be able to perform an incomplete recovery, unless you have a duplicate version of the archive log in another destination.

Assumptions

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

Tasks

Verify that the Control Files are Multiplexed

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. Execute the `$HOME/labs/RMAN_glogin.sh` shell script to set the formatting for all columns selected in queries.

```
$ $HOME/labs/RMAN_glogin.sh
```

```
$
```

- Start SQL*Plus and connect to the CDB root as the SYS user with the SYSDBA privilege.

```
$ sqlplus / AS SYSDBA
...
SQL>
```

- Find out how many control files exist in the database. The query returns the names of two control files (`control01.ctl` and `control02.ctl`), which verifies that the control files are multiplexed. When the CDB was created, DBCA had created two control files. When you use the `CREATE DATABASE` command in SQL*Plus to create a database, you configure the `CONTROL_FILES` parameter to generate two control files and set their names. Multiplexing that ensures security against failure, requires at least one of the control files to be on a separate disk controller, and all of them to be on separate disks.

```
SQL> SELECT name FROM v$controlfile;
NAME
-----
/u01/app/oracle/oradata/ORCL/control01.ctl
/u01/app/oracle/fast_recovery_area/ORCL/control02.ctl
SQL>
```

- View the `CONTROL_FILES` parameter. Notice that the paths to the control files are stored in this parameter. The results below are formatted for easier viewing.

```
SQL> SHOW PARAMETER control_files
NAME          TYPE        VALUE
-----
control_files  string     /u01/app/oracle/oradata/ORCL/control01.ctl,
                           /u01/app/oracle/fast_recovery_area/ORCL/control02.ctl
SQL>
```

Create a Third Control File

For better security, you want to have three copies of the control file. In this section, you add a third control file to the database.

- Create a parameter file (PFILE), which is an ASCII text file that you can edit, from the server parameter file (SPFILE).

```
SQL> CREATE PFILE FROM SPFILE;

File created.
SQL>
```

2. Shut down the database instance in IMMEDIATE mode.

```
SQL> SHUTDOWN IMMEDIATE

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

3. Exit SQL*Plus.

```
SQL> EXIT

...
$
```

4. Create a new directory for the control file.

```
$ mkdir -p /u01/app/oracle/controlfiles_dir/ORCL

$
```

5. Before you edit your PFILE, make a backup copy of it.

```
$ cp $ORACLE_HOME/dbs/initORCL.ora $ORACLE_HOME/dbs/backup_initORCL.ora

$
```

6. Open the PFILE (initORCL.ora) in the vi editor and add a new control file named control03.ctl to the end of the list of control files. Include the path. Be certain not to enter spaces between the single quotes and commas in the control_files= line. Be certain that this line is one continuous line, without line breaks. Save and close the file (:wq).

```
$ vi $ORACLE_HOME/dbs/initORCL.ora

...
*.control_files='/u01/app/oracle/oradata/ORCL/control01.ctl',
 '/u01/app/oracle/fast_recovery_area/ORCL/control02.ctl','/u01/app/oracle/
controlfiles_dir/ORCL/control03.ctl'
...
$
```

7. Copy one of the control files to the /u01/app/oracle/controlfiles_dir/ORCL directory (the one you just created), and name the file as control03.ctl.

```
$ cp /u01/app/oracle/oradata/ORCL/control01.ctl /u01/app/oracle/controlfiles_dir
/ORCL/control03.ctl

$
```

8. Start SQL*Plus and connect to the root container as the SYS user with the SYSDBA privilege. You are connected to an idle instance.

```
$ sqlplus / AS SYSDBA
Connected to an idle instance.
SQL>
```

9. Start up 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>
```

10. View the CONTROL_FILES parameter again.

```
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
SQL>
```

11. Question: Why does the CONTROL_FILES parameter still show only two control files?

Answer: By default, the database instance starts up with the SPFILE. If an SPFILE does not exist, then the instance starts up with a PFILE. In this case, both an SPFILE and PFILE are present, so the SPFILE takes precedence. You configured the PFILE, not the SPFILE. The SPFILE still contains only two references.

12. Recreate the third control file because the current version is now not exactly the same copy of the others.

- Shut down the database instance in IMMEDIATE mode.

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

- Exit SQL*Plus.

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

- c. Issue the `cp` command to recreate `control03.ctl`.

```
$ cp /u01/app/oracle/oradata/ORCL/control01.ctl /u01/app/oracle/
controlfiles_dir/ORCL/control03.ctl

$
```

13. Recreate the SPFILE from the updated PFILE.

- a. Start SQL*Plus and connect to the CDB root as the `SYS` user with the `SYSDBA` privilege. You are connected to an idle instance.

```
$ sqlplus / AS SYSDBA

Connected to an idle instance.
SQL>
```

- b. Create the SPFILE.

```
SQL> CREATE SPFILE FROM PFILE;

File created.

SQL>
```

14. Start up 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>
```

15. View the `CONTROL_FILES` parameter again. The third control file is now included in the list, which indicates that the SPFILE is configured properly. The results below are formatted for easier viewing.

```
SQL> SHOW PARAMETER control_files

NAME          TYPE        VALUE
-----
control_files  string     /u01/app/oracle/oradata/ORCL/control01.ctl,
                           /u01/app/oracle/fast_recovery_area/ORCL/control02.ctl,
                           /u01/app/oracle/controlfiles_dir/ORCL/control03.ctl
SQL>
```

16. Query the `V$CONTROLFILE` view to confirm the number of control files. The result indicates that three control

files are defined.

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

NAME
-----
/u01/app/oracle/oradata/ORCL/control01.ctl
/u01/app/oracle/fast_recovery_area/ORCL/control02.ctl
/u01/app/oracle/controlfiles_dir/ORCL/control03.ctl
SQL>
```

Configure the Size of the Fast Recovery Area

In this section, you review the fast recovery area (FRA) configuration and change its size to 18GB.

1. Question: How can you evaluate the size needed for the FRA?

Answer: The amount of disk space to allocate for the FRA depends on the size and activity levels of your database. As a general rule, the larger the FRA, the more useful it is. Ideally, the FRA should be large enough for copies of your data and control files, as well as for flashback, online redo, and archived logs needed to recover the database with the backups kept based on the retention policy (covered in one of the next practices). In short, the FRA should be at least twice the size of the database so that it can hold one backup and several archived logs.

2. View the DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE initialization parameters.

```
SQL> SHOW PARAMETER db_recovery_file_dest

NAME                      TYPE        VALUE
-----
-
db_recovery_file_dest      string      /u01/app/oracle/fast_recovery_area/ORCL
db_recovery_file_dest_size big integer 12780M
SQL>
```

3. Question: Is the fast recovery area enabled?

Answer: Yes, because the DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE parameters values are not null.

4. Question: Which essential DBA tasks can you perform in the fast recovery area?

Answer: You can change the location and size for the fast recovery area.

5. Question: Does changing the size of the fast recovery area require the database to be restarted?

Answer: No, a restart is not required for this change because the DB_RECOVERY_FILE_DEST_SIZE parameter that you need to modify is dynamic.

6. Change the size of the fast recovery area to **18GB** and set the scope to **both**.

Note: If the archived redo log file destination fills up or cannot be written to, the database will halt. You would then need to remove archived redo log files from the archived redo log file destination so that the database could resume operations. This activity is covered in one of the next practices.

```
SQL> ALTER SYSTEM SET db_recovery_file_dest_size = 18G SCOPE=both;

System altered.
SQL>
```

- View the DB_RECOVERY_FILE_DEST_SIZE initialization parameter again. The result verifies that the size has been set to 18GB.

```
SQL> SHOW PARAMETER db_recovery_file_dest_size
```

NAME	TYPE	VALUE
db_recovery_file_dest_size	big integer	18G

Configure Redo Log Groups

Ensure that there are at least two redo log members in each group. If you are using file system storage, then each member should be distributed on separate disks or controllers so that no single equipment failure impacts an entire log group. The loss of an entire current log group is one of the most serious media failures because it can result in data loss. The loss of a single member of a multi-member log group is trivial and does not affect database operation (other than causing an alert to be published in the alert log). One set of members should be stored in the FRA.

- Query the members for each redo log group. The result shows that there are currently three log groups (1, 2, and 3) and only one member in each group.

```
SQL> SELECT group#, status, member FROM v$logfile;

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

SQL>
```

- Question: Why is it recommended to have three groups when two would be sufficient?
Answer: The Oracle Database server treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next. After all groups have been written to, the Oracle Database server begins overwriting information in the first log group. If the database is configured in ARCHIVELOG mode, the LGWR cannot overwrite data in the first log group if it has not been archived.
- Question: Can multiplexing redo logs impact database performance?
Answer: Multiplexing redo logs may heavily influence database performance because a commit cannot complete until the transaction information has been written to the logs by LGWR. You must place your redo log

files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files. Because only one group is written to at a given time, there is no performance impact in having members from several groups on the same disk.

- Add another member to each redo log group. Name each member as `redonnb.log` where *nn* represents the group number.

```
SQL> ALTER DATABASE ADD LOGFILE MEMBER
  '/u01/app/oracle/fast_recovery_area/ORCL/redo01b.log' TO GROUP 1;

Database altered.

SQL> ALTER DATABASE ADD LOGFILE MEMBER '
/u01/app/oracle/fast_recovery_area/ORCL/redo02b.log' TO GROUP 2;

Database altered.

SQL> ALTER DATABASE ADD LOGFILE MEMBER '
/u01/app/oracle/fast_recovery_area/ORCL/redo03b.log' TO GROUP 3;

Database altered.
SQL>
```

- Verify that the redo log files are now multiplexed. The query result shows that each group has two members and therefore, the redo log files are multiplexed. Observe the `INVALID` status of the newly added redo log members. This status is expected because the new members have not yet been written to by LGWR. When a log switch occurs and the group containing the new member becomes `CURRENT`, the new member's status will change to `null`.

```
SQL> SELECT group#, status, member FROM v$logfile ORDER BY 1, 3;

  GROUP# STATUS    MEMBER
----- -----
  1 INVALID   /u01/app/oracle/fast_recovery_area/ORCL/redo01b.log
  1          /u01/app/oracle/oradata/ORCL/redo01.log
  2 INVALID   /u01/app/oracle/fast_recovery_area/ORCL/redo02b.log
  2          /u01/app/oracle/oradata/ORCL/redo02.log
  3 INVALID   /u01/app/oracle/fast_recovery_area/ORCL/redo03b.log
  3          /u01/app/oracle/oradata/ORCL/redo03.log

6 rows selected.
SQL>
```

- Switch the log files and observe the changes.

- Find out which log group is the current log group. In this example, the query result shows that group 3 is the current group. Your current group may be different.

```
SQL> SELECT group#, members, archived, status FROM v$log;

  GROUP#    MEMBERS  ARC STATUS
-----  -----  -----
      1          2 NO  INACTIVE
      2          2 NO  INACTIVE
      3          2 NO  CURRENT
SQL>
```

- b. Switch the log files three times.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.
SQL>
```

- c. Query the V\$LOGFILE view again. Notice that the switch caused the new members' statuses to change to null.

```
SQL> SELECT group#, status, member FROM v$logfile ORDER BY 1, 3;

  GROUP#  STATUS   MEMBER
-----  -----
      1        /u01/app/oracle/fast_recovery_area/ORCL/redo01b.log
      1        /u01/app/oracle/oradata/ORCL/redo01.log
      2        /u01/app/oracle/fast_recovery_area/ORCL/redo02b.log
      2        /u01/app/oracle/oradata/ORCL/redo02.log
      3        /u01/app/oracle/fast_recovery_area/ORCL/redo03b.log
      3        /u01/app/oracle/oradata/ORCL/redo03.log

6 rows selected.
SQL>
```

- d. Query the V\$LOG view again to learn which log group is now the current group. In this example, the results show that the LGWR is still writing to group 3. Your group may be different. Your statuses may be different too. An INACTIVE status means the log group is no longer needed for database instance recovery.

```
SQL> SELECT group#, members, archived, status FROM v$log;

  GROUP#    MEMBERS   ARC STATUS
-----  -----  -----  -----
        1          2  NO  INACTIVE
        2          2  NO  INACTIVE
        3          2  NO  CURRENT
SQL>
```

- e. Switch the log file.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.
```

- f. Query the V\$LOG view again. The current group has changed to group 1 and the former current group's status is now ACTIVE. Your current group may be different. An ACTIVE status means that the log group is active, but it is not the current log group. It is needed for crash recovery. It may be in use for block recovery. It may or may not be archived.

```
SQL> SELECT group#, members, archived, status FROM v$log;

  GROUP#    MEMBERS   ARC STATUS
-----  -----  -----  -----
        1          2  NO  CURRENT
        2          2  NO  INACTIVE
        3          2  NO  ACTIVE
SQL>
```

- g. Switch the log file again.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.
```

- h. Query the V\$LOG view again. The current group has changed again to group 2 and the status of both the other groups is now ACTIVE. Your current group may be different.

```
SQL> SELECT group#, members, archived, status FROM v$log;

  GROUP#    MEMBERS   ARC STATUS
-----  -----  -----  -----
        1          2  NO  ACTIVE
        2          2  NO  CURRENT
        3          2  NO  ACTIVE
SQL>
```

- i. Question: Can the LGWR background process write to only one member of the CURRENT group in case the other members are missing or damaged?

Answer: Yes, it can. As long as there is one member left in the CURRENT group, LGWR can work.

Configure ARCHIVELOG Mode

Notice in step 6h in the previous section that the archived column (`ARC`) for each redo log group is equal to `NO`. This means that your database is not retaining copies of redo logs to use for database recovery. In the event of a failure, you will lose all your data since your last backup.

In this section, use SQL*Plus to configure your database in `ARCHIVELOG` mode so that redo logs are archived. You do not need to specify a naming convention or a destination for the archived redo log files because you are using a fast recovery area. If you create redundant archive log destinations, you must create the directories if they do not already exist.

1. Shut down the database instance in `IMMEDIATE` mode.

Note: You are currently logged in to the CDB root as the `SYS` user with the `SYSDBA` privilege.

```
SQL> SHUTDOWN IMMEDIATE

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

2. Start the database in `MOUNT` mode.

```
SQL> STARTUP MOUNT

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.
SQL>
```

3. Set the database mode to `ARCHIVELOG`.

```
SQL> ALTER DATABASE ARCHIVELOG;

Database altered.
SQL>
```

4. Open the database.

```
SQL> ALTER DATABASE OPEN;

Database altered.
SQL>
```

5. Shut down the database instance in `IMMEDIATE` mode again.

```
SQL> SHUTDOWN IMMEDIATE

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

6. Start up 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>
```

7. Issue the **ARCHIVE LOG LIST** command to verify that the database is in ARCHIVELOG mode. The result indicates that the database log mode is set to Archive Mode.

Now that your database is in ARCHIVELOG mode, it will continually archive a copy of each online redo log file before reusing it for additional redo data.

ARCn is an optional background process. However, it is crucial to the recovery of a database after the loss of a disk. When an online redo log group gets filled, the Oracle Database server begins writing to the next online redo log group. The process of switching from one online redo log group to another is called a log switch. The ARCn process initiates archiving of the filled log group at every log switch. It automatically archives the online redo log group before the log group can be reused so that all the changes made to the database are preserved. This action enables recovery of the database to the point of failure even if a disk drive is damaged. Remember, this action consumes space on the disk and you must regularly back up older archive logs to some other storage location.

```
SQL> ARCHIVE LOG LIST

Database log mode           Archive Mode
Automatic archival         Enabled
Archive destination        USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 86
Next log sequence to archive 88
Current log sequence       88
SQL>
```

8. Exit SQL*Plus.

```
SQL> EXIT

...
$
```

Configure Redundant Archive Log Destinations

1. Create a new directory named /u01/app/oracle/oradata/ORCL/archive_dir2.

```
$ mkdir /u01/app/oracle/oradata/ORCL/archive_dir2  
$
```

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

```
$ sqlplus / AS SYSDBA  
...  
SQL>
```

3. Set the LOG_ARCHIVE_DEST_1 parameter to the FRA destination (as defined in the USE_DB_RECOVERY_FILE_DEST parameter) and the LOG_ARCHIVE_DEST_2 parameter to the new directory.

```
SQL> ALTER SYSTEM SET log_archive_dest_1='LOCATION=USE_DB_RECOVERY_FILE_DEST'  
SCOPE=both;  
  
System altered.  
  
SQL> ALTER SYSTEM SET  
log_archive_dest_2='LOCATION=/u01/app/oracle/oradata/ORCL/archive_dir2'  
SCOPE=both;  
  
System altered.  
SQL>
```

4. Perform a few log switches.

```
SQL> ALTER SYSTEM SWITCH LOGFILE;  
  
System altered.  
  
SQL> ALTER SYSTEM SWITCH LOGFILE;  
  
System altered.  
  
SQL> ALTER SYSTEM SWITCH LOGFILE;  
  
System altered.  
SQL>
```

5. View the ARCHIVED column value for each log group in the V\$LOG view. In this example, log groups 2 and 3 are archived. Your result may be different.

```
SQL> SELECT group#, members, archived, status FROM v$log;
```

GROUP#	MEMBERS	ARC	STATUS
1	2	NO	CURRENT
2	2	YES	ACTIVE
3	2	YES	INACTIVE

6. Verify that the redundant archive logs are created in both destinations that you just defined by querying the V\$ARCHIVED_LOG view. There are two archives for each log group. The results are formatted for easier viewing.

```
SQL> SELECT name FROM v$archived_log ORDER BY stamp;
```

```
NAME
```

```
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/  
2016_11_16/o1_mf_1_12_d2s600fg_.arc  
/u01/app/oracle/oradata/ORCL/archive_dir2/1_12_927490883.dbf  
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/  
2016_11_16/o1_mf_1_13_d2s601ly_.arc  
/u01/app/oracle/oradata/ORCL/archive_dir2/1_13_927490883.dbf  
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/  
2016_11_16/o1_mf_1_14_d2s603fp_.arc  
/u01/app/oracle/oradata/ORCL/archive_dir2/1_14_927490883.dbf
```

```
6 rows selected.
```

```
SQL>
```

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

```
SQL> EXIT
```

```
...  
$
```

Practice 10-2 Backing up the Control File

Overview

In this practice, you back up your control file to a trace file, and then create a file of SQL commands that can be used to re-create the control file.



Tip:

The loss of a single control file causes the database instance to fail because all control files must be available at all times. However, recovery can be a simple matter of copying one of the other control files. The loss of all control files is slightly more difficult to recover from, but is not usually catastrophic as long as you created a copy of the control file by backing it up to a trace file.

Assumptions

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

You completed [Practice 10-1 Configuring Your Database for Recovery](#).

Tasks

Complete the following steps.

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 CDB root as the `SYSTEM` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>

...
SQL>
```

3. Verify that the control files are multiplexed. The query returns three control files, which indicates that the control files are multiplexed.

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

NAME
-----
/u01/app/oracle/oradata/ORCL/control01.ctl
/u01/app/oracle/fast_recovery_area/ORCL/control02.ctl
/u01/app/oracle/controlfiles_dir/ORCL/control03.ctl
SQL>
```

4. Back up the control file to a trace file. This command writes a SQL script to a trace file where it can be captured and edited to reproduce the control file.

```
SQL> ALTER DATABASE BACKUP controlfile TO trace;

Database altered.
SQL>
```

5. Exit SQL*Plus.

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

6. Navigate to the alert log directory. The alert log contains the name and location of the trace file.

```
$ cd /u01/app/oracle/diag/rdbms/orcl/ORCL/trace
$
```

7. List the files in this directory. Notice that there is the alert log (`alert_ORCL.log`) and many trace files (`TRC` files).

```
$ ls
alert_ORCL.log          ORCL_j000_22943.trc    ORCL_lgwr_11675.trm
ORCL_aqpc_11379.trc    ORCL_j000_22943.trm    ORCL_lgwr_12563.trc
ORCL_aqpc_11379.trm    ORCL_j000_23090.trc    ORCL_lgwr_12563.trm
...
$
```

8. View the end of the alert log and make note of the last trace file created as a backup for the control file. In this example, it is `ORCL_ora_6289.trc`. Your file name will be different.

```
[oracle@12cr2db trace]$ tail alert_ORCL.log
...
Backup controlfile written to trace file
/u01/app/oracle/diag/rdbms/orcl/trace/ORCL_ora_6289.trc
Completed: alter database backup controlfile to trace
$
```

9. View the content of the last generated trace file by using the `cat` command. Make sure to substitute the name of the trace file with your trace file name. Between the lines "`-- Set #1. NORESETLOGS case`" and "`-- Set #2. RESETLOGS case`", select the code from `STARTUP NOMOUNT` to `ALTER SESSION SET CONTAINER = CDB$ROOT`; and copy it to the clipboard.

```
[oracle@12cr2db trace]$ cat  
/u01/app/oracle/diag/rdbms/orcl/ORCL/trace/ORCL_ora_6289.trc  
  
....  
--      Set #1. NORESETLOGS case  
....  
  
STARTUP NOMOUNT  
CREATE CONTROLFILE REUSE DATABASE "ORCL" NORESETLOGS ARCHIVELOG  
  MAXLOGFILES 16  
  MAXLOGMEMBERS 3  
  MAXDATAFILES 1024  
  MAXINSTANCES 8  
  MAXLOGHISTORY 292  
  
LOGFILE  
  GROUP 1 (  
    '/u01/app/oracle/oradata/ORCL redo01.log',  
    '/u01/app/oracle/fast_recovery_area/ORCL redo01b.log'  
  ) SIZE 200M BLOCKSIZE 512,  
  GROUP 2 (  
    '/u01/app/oracle/oradata/ORCL redo02.log',  
    '/u01/app/oracle/fast_recovery_area/ORCL redo02b.log'  
  ) SIZE 200M BLOCKSIZE 512,  
  GROUP 3 (  
    '/u01/app/oracle/oradata/ORCL redo03.log',  
    '/u01/app/oracle/fast_recovery_area/ORCL redo03b.log'  
  ) SIZE 200M BLOCKSIZE 512  
-- STANDBY LOGFILE  
  
DATAFILE  
  '/u01/app/oracle/oradata/ORCL/system01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb1/undotbs01.dbf',  
  '/u01/app/oracle/oradata/ORCL/sysaux01.dbf',  
  '/u01/app/oracle/oradata/ORCL/undotbs01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdbseed/system01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf',  
  '/u01/app/oracle/oradata/ORCL/users01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb1/users01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb1/tbs_app01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb2/system01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb2/sysaux01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb2/undotbs01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb2/users01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb2/tbs_app01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb1/system01.dbf',  
  '/u01/app/oracle/oradata/ORCL/pdb1/sysaux01.dbf'  
  
CHARACTER SET AL32UTF8
```

```
;  
RECOVER DATABASE  
ALTER SYSTEM ARCHIVE LOG ALL;  
ALTER DATABASE OPEN;  
ALTER PLUGGABLE DATABASE ALL OPEN;  
ALTER TABLESPACE TEMP ADD TEMPFILE '/u01/app/oracle/oradata/ORCL/temp01.dbf'  
SIZE 254803968 REUSE AUTOEXTEND ON NEXT 655360 MAXSIZE 32767M;  
ALTER SESSION SET CONTAINER = PDB$SEED;  
ALTER TABLESPACE TEMP ADD TEMPFILE  
'/u01/app/oracle/oradata/ORCL/pdbseed/temp012016-11-09_20-22-01-114-PM.dbf'  
SIZE 67108864 REUSE AUTOEXTEND ON NEXT 655360 MAXSIZE 32767M;  
ALTER SESSION SET CONTAINER = PDB1;  
ALTER TABLESPACE TEMP ADD TEMPFILE  
'/u01/app/oracle/oradata/ORCL/PDB1/temp01.dbf' REUSE;  
ALTER SESSION SET CONTAINER = CDB$ROOT;  
-- End of tempfile additions  
--  
-- Set #2. RESETLOGS case  
...  
...
```

10. Open gedit (On your desktop, select **Applications**, **Accessories**, and then **gedit Text Editor**) and paste the code into a new file. In gedit, save this code to the /home/oracle directory (the default) as ControlFileBackup.sql.
11. Question: Which command would allow the recreation of the control files in case of a complete loss of the control files?
Answer: In the case where all control files are lost, the CREATE CONTROLFILE command in the trace file would recreate the missing control files with the right information, keeping the database file structure in terms of data files, redo log files, and other database attributes (ARCHIVELOG, maximum settings).
12. Question: How would you execute the command?
Answer: After trimming the trace file by keeping all commands from the STARTUP NOMOUNT up to ALTER SESSION SET CONTAINER = CDB\$ROOT;, you would execute the file as a SQL script.
13. Question: Are the data files, temp files, and control files that structure the ORCL database included in the SQL script?
Answer: Yes, they are included. All data and temp files of the different containers (the CDB root, CDB seed, PDB1, and so on) and the multiplexed redo log files are present.
14. Question: Which other attributes structure the ORCL database?
Answer: The ARCHIVELOG mode, the character set, and the name of the CDB.
15. Question: Why are there two cases:
-- Set #1. NORESETLOGS case
-- Set #2. RESETLOGS case
Answer: The first case from the STARTUP NOMOUNT to the ALTER SESSION SET CONTAINER = CDB\$ROOT provides a script to execute a complete database recovery. Use this only if the current versions of all online logs are available. The second case provides a script to execute an incomplete database recovery. Use this only if online logs are damaged. The contents of online logs will be lost and all backups will be invalidated.

16. Question: When would you have to regenerate the trace file from the current control files?
Answer: Because the control file changes after each data file or redo log file change (adding, removing, resizing) or database attribute change (ARCHIVELOG), you would have to redo the back up of your control file to a trace file.
17. Close the terminal window and gedit.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Practice 10-3 Configuring Automatic Backups of the Control File and SPFILE

Overview

In this practice you use Recovery Manager (RMAN) to configure automatic backups of the control file and server parameter file (SPFILE) when a backup of the database is made and when there is a structural change to the database.

Assumptions

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

You completed [Practice 10-1 Configuring Your Database for Recovery](#).

Tasks

Complete the following steps.

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 Recovery Manager and connect to the CDB root (target database) as the `SYS` user.

```
$ rman target /

Recovery Manager: Release 12.2.0.1.0 - Production on Wed Jan 11 18:01:11 2017
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.
connected to target database: ORCL (DBID=1458832693)
RMAN>
```

3. Show all RMAN settings. Notice the line `CONFIGURE CONTROLFILE AUTOBACKUP ON; # default` (sixth line down).

```
RMAN> SHOW ALL;

using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR
LOAD TRUE ; # default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/u01/app/oracle/product/12.2.0/dbhome_1/dbs/snapcf_ORCL.f'; # default

RMAN>
```

4. Question: Does RMAN automatically backup the control file and server parameter file (SPFILE) with every backup and database structural change?
Answer: Yes, it does because the CONTROLFILE AUTOBACKUP attribute is set to ON by default.
5. To make a test, set the CONTROLFILE AUTOBACKUP attribute to OFF and check whether this is permanent from one RMAN session to another.

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP OFF;

new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
new RMAN configuration parameters are successfully stored
RMAN>
```

6. Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.
$
```

7. Start Recovery Manager again and connect to the CDB root (target database) as the SYS user.

```
$ rman target /
```

```
Recovery Manager: Release 12.2.0.1.0 - Production on Wed Jan 11 18:01:11 2017
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.
connected to target database: ORCL (DBID=1458832693)
RMAN>
```

8. Show all RMAN settings again. Notice in the code that the CONTROLFILE AUTOBACKUP attribute is still set to OFF; therefore, the change is permanent from one RMAN session to another.

```
RMAN> SHOW ALL;
```

```
using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR
LOAD TRUE ; # default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/u01/app/oracle/product/12.2.0/dbhome_1/dbs/snapcf_ORCL.f'; # default

RMAN>
```

9. Set the CONTROLFILE AUTOBACKUP attribute back to ON.

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

```
old RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP OFF;
new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored
RMAN>
```

10. Question: Will a backup operation back up all control files or only one of the multiplexed control files?
Answer: It will back up only one of the multiplexed control files because all control files in a database are identical.

11. Exit RMAN and close the terminal window.

```
RMAN> EXIT;
```

Practice 10-4 Creating a Whole Database Backup

Overview

In this practice, you use Recovery Manager to back up your entire database, including the archived redo log files, the SPFILE, and the control files. The backup should be the base for an incremental backup strategy.

Note: The results in this practice are formatted for easier viewing. Don't worry if your results look slightly different.

Assumptions

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

You completed the following practices:

- Practice 10-1 Configuring Your Database for Recovery
- Practice 10-3 Configuring Automatic Backups of the Control File and SPFILE

Tasks

Complete the following steps.

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. Execute the `$HOME/labs/RMAN_setup.sh` shell script to create `PDB1` and `PDB2` with different application tables.

```
$ $HOME/labs/RMAN_setup.sh

...
$
```

3. Start Oracle Recovery Manager (RMAN) and connect to the CDB root (`ORCL` is the target database) as the `SYS` user.

```
$ rman target /
...
connected to target database: ORCL (DBID=1458832693)
RMAN>
```

4. Report the structure of the CDB in terms of PDBs, tablespaces, and data files (permanent and temporary). Your file numbers will be different than those shown below.

```
RMAN> REPORT schema;

using target database control file instead of recovery catalog
Report of database schema for database with db_unique_name ORCL

List of Permanent Datafiles
=====
File Size(MB) Tablespace          RB segs Datafile Name
-----
1    830      SYSTEM             YES
/u01/app/oracle/oradata/ORCL/system01.dbf
3    950      SYSAUX            NO
/u01/app/oracle/oradata/ORCL/sysaux01.dbf
4    170      UNDOTBS1          YES
/u01/app/oracle/oradata/ORCL/undotbs01.dbf
5    250      PDB$SEED:SYSTEM   NO
/u01/app/oracle/oradata/ORCL/pdbseed/system01.dbf
6    350      PDB$SEED:SYSAUX  NO
/u01/app/oracle/oradata/ORCL/pdbseed/sysaux01.dbf
7    5       USERS              NO
/u01/app/oracle/oradata/ORCL/users01.dbf
8    100     PDB$SEED:UNDOTBS1 NO
/u01/app/oracle/oradata/ORCL/pdbseed/undotbs01.dbf
42   250     PDB1:SYSTEM        YES
/u01/app/oracle/oradata/ORCL/pdb1/system01.dbf
43   350     PDB1:SYSAUX       NO
/u01/app/oracle/oradata/ORCL/pdb1/sysaux01.dbf
44   100     PDB1:UNDOTBS1     YES
/u01/app/oracle/oradata/ORCL/pdb1/undotbs01.dbf
45   250     PDB1:USERS        NO
/u01/app/oracle/oradata/ORCL/pdb1/users01.dbf
46   250     PDB2:SYSTEM        YES
/u01/app/oracle/oradata/ORCL/pdb2/system01.dbf
47   350     PDB2:SYSAUX       NO
/u01/app/oracle/oradata/ORCL/pdb2/sysaux01.dbf
48   100     PDB2:UNDOTBS1     YES
/u01/app/oracle/oradata/ORCL/pdb2/undotbs01.dbf
49   250     PDB2:USERS        NO
/u01/app/oracle/oradata/ORCL/pdb2/users01.dbf
50   800     PDB1:TBS_APP      NO
/u01/app/oracle/oradata/ORCL/pdb1/tbs_app01.dbf
51   800     PDB2:TBS_APP      NO
/u01/app/oracle/oradata/ORCL/pdb2/tbs_app01.dbf

List of Temporary Files
=====
File Size(MB) Tablespace          Maxsize(MB) Tempfile Name
-----
1    131      TEMP              32767
/u01/app/oracle/oradata/ORCL/temp01.dbf
2    64       PDB$SEED:TEMP     32767
/u01/app/oracle/oradata/ORCL/pdbseed/
temp012016-12-20_01-33-16-961-AM.dbf
5    64       PDB1:TEMP         32767
/u01/app/oracle/oradata/ORCL/pdb1/
temp012016-12-20_01-33-16-961-AM.dbf
```

```
8      64          PDB2:TEMP           32767
/u01/app/oracle/oradata/ORCL/pdb2/
temp012016-12-20_01-33-16-961-AM.dbf

RMAN>
```

5. Back up the whole database and the archive log files. Your results will be different than the results shown below; for example, the piece handle names will be different.

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;

Starting backup at 23-JAN-17
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=288 device type=DISK
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=47 RECID=1 STAMP=934060805
input archived log thread=1 sequence=48 RECID=3 STAMP=934060809
input archived log thread=1 sequence=49 RECID=5 STAMP=934060814
input archived log thread=1 sequence=50 RECID=7 STAMP=934062225
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_23/o1_mf_annnn_TAG20170123T214345_d8dy8kvw_.bkp
tag=TAG20170123T214345 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17

Starting backup at 23-JAN-17
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00003 name=/u01/app/oracle/oradata/ORCL/sysaux01.dbf
input datafile file number=00001 name=/u01/app/oracle/oradata/ORCL/system01.dbf
input datafile file number=00004 name=/u01/app/oracle/oradata/ORCL/undotbs01.dbf
input datafile file number=00007 name=/u01/app/oracle/oradata/ORCL/users01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_23/o1_mf_nnndf_TAG20170123T214347_d8dy8mb6_.bkp
tag=TAG20170123T214347 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:26
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00050 name=/u01/app/oracle/oradata/ORCL/pdb1/
tbs_app01.dbf
input datafile file number=00043 name=/u01/app/oracle/oradata/ORCL/pdb1/
sysaux01.dbf
input datafile file number=00042 name=/u01/app/oracle/oradata/ORCL/pdb1/
system01.dbf
input datafile file number=00045 name=/u01/app/oracle/oradata/ORCL/pdb1/
users01.dbf
```

```
input datafile file number=00044 name=/u01/app/oracle/oradata/ORCL/pdb1/
undotbs01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy9fnf_.bkp tag=TAG20170123T214347
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00051 name=/u01/app/oracle/oradata/ORCL/
pdb2/tbs_app01.dbf
input datafile file number=00047 name=/u01/app/oracle/oradata/ORCL/
pdb2/sysaux01.dbf
input datafile file number=00046 name=/u01/app/oracle/oradata/ORCL/
pdb2/system01.dbf
input datafile file number=00049 name=/u01/app/oracle/oradata/ORCL/
pdb2/users01.dbf
input datafile file number=00048 name=/u01/app/oracle/oradata/ORCL/
pdb2/undotbs01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy9ntg_.bkp tag=TAG20170123T214347
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00006 name=/u01/app/oracle/oradata/ORCL/
pbseed/sysaux01.dbf
input datafile file number=00005 name=/u01/app/oracle/oradata/ORCL/
pbseed/system01.dbf
input datafile file number=00008 name=/u01/app/oracle/oradata/ORCL/
pbseed/undotbs01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
440E054176A52B5FE0530F11ED0A326A/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy9vmb_.bkp tag=TAG20170123T214347
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 23-JAN-17

Starting backup at 23-JAN-17
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=51 RECID=9 STAMP=934062274
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_23/o1_mf_annnn_TAG20170123T214434_d8dyb2v6_.bkp
tag=TAG20170123T214434 comment=NONE
```

```
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17

Starting Control File and SPFILE Autobackup at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/
2017_01_23/o1_mf_s_934062276_d8dyb4bq_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 23-JAN-17
RMAN>
```

6. Question: Do you have to shut down the database to back it up?

Answer: No, as long as the database is in ARCHIVELOG mode, the backup can take place while the database is opened. This is a hot backup (or online backup). A cold backup (or offline backup) is a backup completed while the database is closed and is required if the database is in NOARCHIVELOG mode.

7. Question: Are hot backups consistent?

Answer: Online backups are inconsistent because with the database opened, there is no guarantee that the data files are synchronized with the control files. However, offline backups taken while the database is not opened are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

8. Question: What would allow hot backups (inconsistent backups) to perform a complete database recovery?

Answer: During a complete recovery, restored online backups are recovered until the current SCN is matched, with the use of the archive log files and online redo log files.

9. Question: Did the backup include the SPFILE and control files?

Answer: Yes. This is the last operation completed at the end of the backup command.

```
Starting Control File and SPFILE Autobackup at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/
2017_01_23/o1_mf_s_934062276_d8dyb4bq_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 23-JAN-17
```

10. Question: Does the complete operation create a single backup set?

Answer: No. The operation creates seven backup sets, as listed in the results in step 5.

- Four backup sets including data files (one for each of the containers): CDB root, CDB seed, PDB1, PDB2

```
(Root)
piece
handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy8mb6_.bkp tag=TAG20170123T214347
comment=NONE

(PDB1)
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy9fnf_.bkp tag=TAG20170123T214347
comment=NONE

(PDB2)
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy9ntg_.bkp tag=TAG20170123T214347
comment=NONE

(Seed)
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
440E054176A52B5FE0530F11ED0A326A/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy9vmb_.bkp tag=TAG20170123T214347
comment=NONE
```

- One backup set for the archive log files before the data files back up and another backup set after the data files back up.

```
(Before)
Starting backup at 23-JAN-17
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=288 device type=DISK
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=47 RECID=1 STAMP=934060805
input archived log thread=1 sequence=48 RECID=3 STAMP=934060809
input archived log thread=1 sequence=49 RECID=5 STAMP=934060814
input archived log thread=1 sequence=50 RECID=7 STAMP=934062225
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_23/o1_mf_annnn_TAG20170123T214345_d8dy8kvw_.bkp
tag=TAG20170123T214345
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17
```

(After)

```

Starting backup at 23-JAN-17
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=51 RECID=9 STAMP=934062274
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_23/o1_mf_annnn_TAG20170123T214434_d8dyb2v6_.bkp
tag=TAG20170123T214434
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17

```

- One backup set for the SPFILE and control files.

```

Starting Control File and SPFILE Autobackup at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/
2017_01_23/o1_mf_s_934062276_d8dyb4bq_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 23-JAN-17

```

11. List the backup sets. Look for Piece Name in the results for each backup set.

```
RMAN> LIST BACKUP;
```

List of Backup Sets

BS	Key	Size	Device	Type	Elapsed Time	Completion Time
1		23.01M	DISK		00:00:00	23-JAN-17
			BP Key: 1	Status: AVAILABLE	Compressed: NO	Tag: TAG20170123T214345
			Piece Name:	/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/		
				2017_01_23/o1_mf_annnn_TAG20170123T214345_d8dy8kvw_.bkp		

List of Archived Logs in backup set 1

Thrd	Seq	Low SCN	Low Time	Next SCN	Next Time
1	47	3697321	23-JAN-17	3699121	23-JAN-17
1	48	3699121	23-JAN-17	3699124	23-JAN-17
1	49	3699124	23-JAN-17	3699131	23-JAN-17
1	50	3699131	23-JAN-17	3714445	23-JAN-17

BS Key Type LV Size Device Type Elapsed Time Completion Time

2	Full	1.61G	DISK		00:00:18	23-JAN-17
	BP Key: 2	Status: AVAILABLE	Compressed: NO	Tag: TAG20170123T214347		
	Piece Name:	/u01/app/oracle/fast_recovery_area/ORCL/ORCL/				
	backupset/2017_01_23/o1_mf_nnndf_TAG20170123T214347_d8dy8mb6_.bkp					

List of Datafiles in backup set 2

File	LV	Type	Ckp	SCN	Ckp	Time	Abs	Fuz	SCN	Sparse	Name
1		Full	3714454			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/system01.dbf
3		Full	3714454			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/sysaux01.dbf
4		Full	3714454			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/undotbs01.dbf
7		Full	3714454			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/users01.dbf

BS Key Type LV Size Device Type Elapsed Time Completion Time

BS Key	Type	LV	Size	Device	Type	Elapsed Time	Completion Time
3	Full		486.23M	DISK		00:00:03	23-JAN-17
	BP Key: 3	Status: AVAILABLE		Compressed: NO		Tag: TAG20170123T214347	
	Piece Name:	/u01/app/oracle/fast_recovery_area/ORCL/ORCL/46CAD91DBF494767E053F511ED0A6A98/backupset/2017_01_23/o1_mf_nnndf_TAG20170123T214347_d8dy9fnf_.bkp					

List of Datafiles in backup set 3

Container ID: 5, PDB Name: PDB1

File	LV	Type	Ckp	SCN	Ckp	Time	Abs	Fuz	SCN	Sparse	Name
42		Full	3714463			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb1/system01.dbf
43		Full	3714463			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb1/sysaux01.dbf
44		Full	3714463			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb1/undotbs01.dbf
45		Full	3714463			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb1/users01.dbf
50		Full	3714463			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb1/tbs_app01.dbf

BS Key Type LV Size Device Type Elapsed Time Completion Time

BS Key	Type	LV	Size	Device	Type	Elapsed Time	Completion Time
4	Full		486.25M	DISK		00:00:03	23-JAN-17
	BP Key: 4	Status: AVAILABLE		Compressed: NO		Tag: TAG20170123T214347	
	Piece Name:	/u01/app/oracle/fast_recovery_area/ORCL/ORCL/46CAD91DBF4E4767E053F511ED0A6A98/backupset/2017_01_23/o1_mf_nnndf_TAG20170123T214347_d8dy9ntg_.bkp					

List of Datafiles in backup set 4

Container ID: 8, PDB Name: PDB2

File	LV	Type	Ckp	SCN	Ckp	Time	Abs	Fuz	SCN	Sparse	Name
46		Full	3714478			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb2/system01.dbf
47		Full	3714478			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb2/sysaux01.dbf
48		Full	3714478			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb2/undotbs01.dbf
49		Full	3714478			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb2/users01.dbf
51		Full	3714478			23-JAN-17				NO	/u01/app/oracle/oradata/ORCL/pdb2/tbs_app01.dbf

```

BS Key Type LV Size      Device Type Elapsed Time Completion Time
----- -----
5     Full   513.96M   DISK        00:00:01    23-JAN-17
      BP Key: 5  Status: AVAILABLE  Compressed: NO  Tag: TAG20170123T214347
      Piece Name: /u01/app/oracle/fast_recovery_area/ORCL/ORCL/
440E054176A52B5FE0530F11ED0A326A/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T214347_d8dy9vmb_.bkp
      List of Datafiles in backup set 5
      Container ID: 2, PDB Name: PDB$SEED
      File LV Type Ckp SCN      Ckp Time Abs Fuz SCN Sparse Name
----- -----
5     Full 1441711 20-DEC-16          NO   /u01/app/oracle/oradata/
ORCL/pdbseed/system01.dbf
6     Full 1441711 20-DEC-16          NO   /u01/app/oracle/oradata/
ORCL/pdbseed/sysaux01.dbf
8     Full 1441711 20-DEC-16          NO   /u01/app/oracle/oradata/
ORCL/pdbseed/undotbs01.dbf

BS Key Size      Device Type Elapsed Time Completion Time
----- -----
6     47.00K   DISK        00:00:00    23-JAN-17
      BP Key: 6  Status: AVAILABLE  Compressed: NO  Tag: TAG20170123T214434
      Piece Name: /u01/app/oracle/fast_recovery_area/ORCL/ORCL
backupset/2017_01_23/o1_mf_annnn_TAG20170123T214434_d8dyb2v6_.bkp

      List of Archived Logs in backup set 6
      Thrd Seq      Low SCN      Low Time  Next SCN      Next Time
----- -----
1     51       3714445  23-JAN-17  3714487  23-JAN-17

BS Key Type LV Size      Device Type Elapsed Time Completion Time
----- -----
7     Full   17.94M   DISK        00:00:00    23-JAN-17
      BP Key: 7  Status: AVAILABLE  Compressed: NO  Tag: TAG20170123T214436
      Piece Name: /u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/
2017_01_23/o1_mf_s_934062276_d8dyb4bq_.bkp
      SPFILE Included: Modification time: 23-JAN-17
      SPFILE db_unique_name: ORCL
      Control File Included: Ckp SCN: 3714496      Ckp time: 23-JAN-17

RMAN>

```

12. Exit RMAN.

```

RMAN> EXIT
$
```

13. Verify that the files are stored on disk in the FRA.

```

$ cd /u01/app/oracle/fast_recovery_area/ORCL/ORCL
$ ls -ltR
```

```
total 28
drwxr-x--- 4 oracle oinstall 4096 Jan 23 21:44 archivelog
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:44 autobackup
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:44
440E054176A52B5FE0530F11ED0A326A
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:44
46CAD91DBF4E4767E053F511ED0A6A98
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:44
46CAD91DBF494767E053F511ED0A6A98
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:43 backupset
drwxr-x--- 2 oracle oinstall 4096 Dec 20 01:32 onlinelog

./archivelog:
total 8
drwxr-x--- 2 oracle oinstall 4096 Jan 23 22:02 2017_01_23
drwxr-x--- 2 oracle oinstall 4096 Jan 18 15:24 2017_01_18

./archivelog/2017_01_23:
total 203308
-rw-r---- 1 oracle oinstall 183780352 Jan 23 22:02 ol_mf_1_52_d8dzclfk_.arc
-rw-r---- 1 oracle oinstall      46080 Jan 23 21:44 ol_mf_1_51_d8dyb2op_.arc
-rw-r---- 1 oracle oinstall    21400064 Jan 23 21:43 ol_mf_1_50_d8dy8jsq_.arc
-rw-r---- 1 oracle oinstall      4608 Jan 23 21:20 ol_mf_1_49_d8dwwg62_.arc
-rw-r---- 1 oracle oinstall      1024 Jan 23 21:20 ol_mf_1_48_d8dww9n7_.arc
-rw-r---- 1 oracle oinstall    2721792 Jan 23 21:20 ol_mf_1_47_d8dww5xo_.arc

./archivelog/2017_01_18:
total 0

./autobackup:
total 4
drwxr-x--- 2 oracle oinstall 4096 Jan 23 21:44 2017_01_23

./autobackup/2017_01_23:
total 18408
-rw-r---- 1 oracle oinstall 18825216 Jan 23 21:44
ol_mf_s_934062276_d8dyb4bq_.bkp

./440E054176A52B5FE0530F11ED0A326A:
total 4
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:44 backupset

./440E054176A52B5FE0530F11ED0A326A/backupset:
total 4
drwxr-x--- 2 oracle oinstall 4096 Jan 23 21:44 2017_01_23

./440E054176A52B5FE0530F11ED0A326A/backupset/2017_01_23:
total 526824
-rw-r---- 1 oracle oinstall 538935296
Jan 23 21:44 ol_mf_nnndf_TAG20170123T214347_d8dy9vmb_.bkp

./46CAD91DBF4E4767E053F511ED0A6A98:
total 4
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:44 backupset

./46CAD91DBF4E4767E053F511ED0A6A98/backupset:
```

```
total 4
drwxr-x--- 2 oracle oinstall 4096 Jan 23 21:44 2017_01_23

./46CAD91DBF4E4767E053F511ED0A6A98/backupset/2017_01_23:
total 498420
-rw-r----- 1 oracle oinstall 509878272
Jan 23 21:44 ol_mf_nnndf_TAG20170123T214347_d8dy9ntg_.bkp

./46CAD91DBF494767E053F511ED0A6A98:
total 4
drwxr-x--- 3 oracle oinstall 4096 Jan 23 21:44 backupset

./46CAD91DBF494767E053F511ED0A6A98/backupset:
total 4
drwxr-x--- 2 oracle oinstall 4096 Jan 23 21:44 2017_01_23

./46CAD91DBF494767E053F511ED0A6A98/backupset/2017_01_23:
total 498404
-rw-r----- 1 oracle oinstall 509861888 Jan 23 21:44
ol_mf_nnndf_TAG20170123T214347_d8dy9fnf_.bkp

./backupset:
total 4
drwxr-x--- 2 oracle oinstall 4096 Jan 23 21:44 2017_01_23

./backupset/2017_01_23:
total 1709080
-rw-r----- 1 oracle oinstall      48640 Jan 23 21:44
ol_mf_annnn_TAG20170123T214434_d8dyb2v6_.bkp
-rw-r----- 1 oracle oinstall 1724194816 Jan 23 21:44
ol_mf_nnndf_TAG20170123T214347_d8dy8mb6_.bkp
-rw-r----- 1 oracle oinstall 24130048 Jan 23 21:43
ol_mf_annnn_TAG20170123T214345_d8dy8kvw_.bkp

./onlinelog:
total 0
$
```

- a. Question: Where are the backups of control files and SPFILE located?
Answer: They are created in the `autobackup` subdirectory.

- b. Question: How are backups deleted?
Answer: Space management in the FRA is governed by a backup retention policy. A retention policy determines when files are obsolete, which means that they are no longer needed to meet your data recovery objectives. The Oracle Database server automatically manages this storage by deleting files that are no longer needed.

14. Display the backup retention policy.

- a. Start RMAN and connect to the CDB root as the `SYS` user.

```
$ rman target /
...
RMAN>
```

- b. Issue the SHOW RETENTION POLICY command. The policy is CONFIGURE RETENTION POLICY TO REDUNDANCY 1, as shown below.

```
RMAN> SHOW RETENTION POLICY;
```

```
using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
RMAN>
```

15. Question: How does Oracle determine when files are obsolete?

Answer: There are two retention policy parameters which are mutually exclusive:

- If a retention policy is enabled with CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 5 DAYS, the window stretches from the current time (SYSDATE) to the point of recoverability, which is the earliest date to which you want to recover. The point of recoverability is SYSDATE - integer days in the past. Use this setting to restore or recover dropped tablespaces or data files.
- If a retention policy is enabled with CONFIGURE RETENTION POLICY TO REDUNDANCY r , then RMAN skips backups only if at least n backups of an identical file exist on the specified device, where $n=r+1$ (default is 1).
- RMAN automatically deletes obsolete backup sets and copies in the FRA when space is needed.

16. Back up the database and archive logs as image copies. At the same time, free space in the FRA by deleting the archive log files once they are backed up.

- a. Perform the backup.

```
RMAN> BACKUP AS COPY DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

```
Starting backup at 23-JAN-17
current log archived
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=261 device type=DISK
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=52 RECID=11 STAMP=934063348
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_23/o1_mf_1_52_d8f099f1_.arc RECID=15 STAMP=934064298
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=53 RECID=13 STAMP=934064296
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_23/o1_mf_1_53_d8f09dj3_.arc RECID=16 STAMP=934064300
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=50 RECID=7 STAMP=934062225
```

```
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_23/o1_mf_1_50_d8f09fml_.arc RECID=17 STAMP=934064301
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=47 RECID=1 STAMP=934060805
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_23/o1_mf_1_47_d8f09gp9_.arc RECID=18 STAMP=934064302
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_47_d8dww5xo_.arc RECID=1 STAMP=934060805
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=51 RECID=9 STAMP=934062274
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_23/o1_mf_1_51_d8f09hsm_.arc RECID=19 STAMP=934064303
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=49 RECID=5 STAMP=934060814
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_23/o1_mf_1_49_d8f09jwf_.arc RECID=20 STAMP=934064304
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=48 RECID=3 STAMP=934060809
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_48_d8f09kz7_.arc RECID=21 STAMP=934064306
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:02
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_48_d8dww9n7_.arc RECID=3 STAMP=934060809
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_49_d8dwwg62_.arc RECID=5 STAMP=934060814
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_50_d8dy8jsq_.arc RECID=7 STAMP=934062225
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_51_d8dyb2op_.arc RECID=9 STAMP=934062274
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_52_d8dzclfk_.arc RECID=11 STAMP=934063348
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_53_d8f09818_.arc RECID=13 STAMP=934064296
Finished backup at 23-JAN-17

Starting backup at 23-JAN-17
using channel ORA_DISK_1
channel ORA_DISK_1: starting datafile copy
input datafile file number=00003 name=/u01/app/oracle/oradata/ORCL/
sysaux01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/datafile/
o1_mf_sysaux_d8f09mj4_.dbf tag=TAG20170123T221827 RECID=4 STAMP=934064320
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting datafile copy
```

```
input datafile file number=00001
name=/u01/app/oracle/oradata/ORCL/system01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/datafile/
o1_mf_system_d8f0b2yp_.dbf tag=TAG20170123T221827 RECID=5 STAMP=934064332
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting datafile copy
input datafile file number=00050 name=/u01/app/oracle/oradata/ORCL/pdb1/
tbs_app01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/datafile/o1_mf_tbs_app_d8f0b111_.dbf
tag=TAG20170123T221827 RECID=6 STAMP=934064346
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:16
channel ORA_DISK_1: starting datafile copy
input datafile file number=00051 name=/u01/app/oracle/oradata/ORCL/pdb2/
tbs_app01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/datafile/o1_mf_tbs_app_d8f0c28h_.dbf
tag=TAG20170123T221827 RECID=7 STAMP=934064362
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:15
channel ORA_DISK_1: starting datafile copy
input datafile file number=00043 name=/u01/app/oracle/oradata/ORCL/pdb1/
sysaux01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/datafile/o1_mf_sysaux_d8f0ckd5_.dbf
tag=TAG20170123T221827 RECID=8 STAMP=934064374
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00047 name=/u01/app/oracle/oradata/ORCL/pdb2/
sysaux01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/datafile/o1_mf_sysaux_d8f0crhn_.dbf
tag=TAG20170123T221827 RECID=9 STAMP=934064381
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:07
channel ORA_DISK_1: starting datafile copy
input datafile file number=00006 name=/u01/app/oracle/oradata/ORCL/
pdbseed/sysaux01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
440E054176A52B5FE0530F11ED0A326A/datafile/o1_mf_sysaux_d8f0czlk_.dbf
tag=TAG20170123T221827 RECID=10 STAMP=934064385
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00005 name=/u01/app/oracle/oradata/ORCL/
pdbseed/system01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORC
440E054176A52B5FE0530F11ED0A326A/datafile/o1_mf_system_d8f0d2or_.dbf
tag=TAG20170123T221827 RECID=11 STAMP=934064389
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00042 name=/u01/app/oracle/oradata/ORCL/pdb1/
system01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/datafile/o1_mf_system_d8f0d5sn_.dbf
tag=TAG20170123T221827 RECID=12 STAMP=934064392
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
```

```
input datafile file number=00045 name=/u01/app/oracle/oradata/ORCL/pdb1/
users01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/datafile/o1_mf_users_d8f0d8ww_.dbf
tag=TAG20170123T221827 RECID=13 STAMP=934064395
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00046 name=/u01/app/oracle/oradata/ORCL/pdb2/
system01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/datafile/o1_mf_system_d8f0dd13_.dbf
tag=TAG20170123T221827 RECID=14 STAMP=934064398
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00049 name=/u01/app/oracle/oradata/ORCL/pdb2/
users01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/datafile/o1_mf_users_d8f0dh4b_.dbf
tag=TAG20170123T221827 RECID=15 STAMP=934064401
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00008 name=/u01/app/oracle/oradata/ORCL/pdbseed/
undotbs01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
440E054176A52B5FE0530F11ED0A326A/datafile/o1_mf_undotbs1_d8f0dl7c_.dbf
tag=TAG20170123T221827 RECID=16 STAMP=934064402
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile copy
input datafile file number=00044 name=/u01/app/oracle/oradata/ORCL/pdb1/
undotbs01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/datafile/o1_mf_undotbs1_d8f0dmc9_.dbf
tag=TAG20170123T221827 RECID=17 STAMP=934064404
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00048 name=/u01/app/oracle/oradata/ORCL/pdb2/
undotbs01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/datafile/o1_mf_undotbs1_d8f0dpgo_.dbf
tag=TAG20170123T221827 RECID=18 STAMP=934064407
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile copy
input datafile file number=00004
name=/u01/app/oracle/oradata/ORCL/undotbs01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/datafile/
o1_mf_undotbs1_d8f0ds1q_.dbf tag=TAG20170123T221827 RECID=19
STAMP=934064410
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: starting datafile copy
input datafile file number=00007
name=/u01/app/oracle/oradata/ORCL/users01.dbf
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/datafile/
o1_mf_users_d8f0dtvn_.dbf tag=TAG20170123T221827 RECID=20 STAMP=934064410
channel ORA_DISK_1: datafile copy complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17
```

```

Starting backup at 23-JAN-17
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log copy
input archived log thread=1 sequence=54 RECID=22 STAMP=934064411
output file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_23/01_mf_1_54_d8f0dvwz_.arc RECID=24 STAMP=934064411
channel ORA_DISK_1: archived log copy complete, elapsed time: 00:00:01
channel ORA_DISK_1: deleting archived log(s)
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/01_mf_1_54_d8f0dqv9_.arc RECID=22 STAMP=934064411
Finished backup at 23-JAN-17

Starting Control File and SPFILE Autobackup at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/
2017_01_23/01_mf_s_934064413_d8f0dxb3_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 23-JAN-17

RMAN>

```

- b. Question: What would you do if an error message like the following one occurs?

```

RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03002: failure of backup plus archivelog command at 01/23/2017
11:05:08
ORA-19809: limit exceeded for recovery files
ORA-19804: cannot reclaim 67108864 bytes disk space from 19327352832 bytes
limit

```

Answer: Increase the db_recovery_file_dest_size parameter value to 30G by issuing the following command:

```
RMAN> ALTER SYSTEM SET db_recovery_file_dest_size = 30G SCOPE=both;
```

- c. Question: What is the advantage of creating backups as image copies?

Answer: The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy, only the file or files need to be retrieved from your backup location. With backup sets, the entire backup set must be retrieved from your backup location before you extract the file or files that are needed.

- d. Question: What is the advantage of creating backups as backup sets?

Answer: The advantage of creating backups as backup sets is better space usage. In most databases, 20% or more of the data blocks are empty blocks. Image copies back up every data block, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems, the advantages of backup sets outweigh the advantages of image copies.

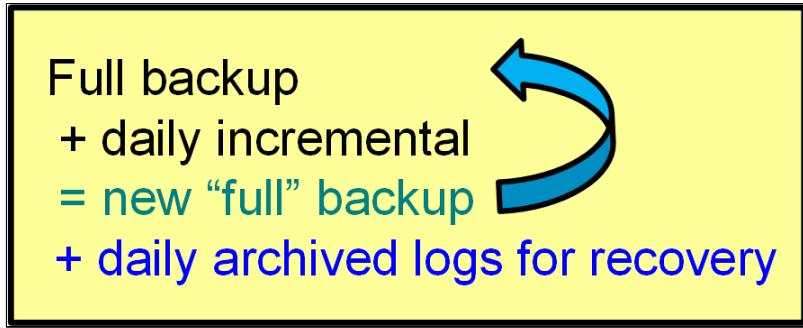
- e. Question: How many image copies of the data files are created?

Answer: There are 15 image copies, one image copy for each data file in the CDB, PDBs included.

17. Consider that you now must schedule a backup strategy to protect your data and provide efficient recoverability to any point in time.

Note: Oracle Enterprise Cloud Control recommends an Oracle-backup strategy schedule that provides efficient

recoverability to any point in the preceding 24 hours, and possibly as far back as 48 hours, depending on when the last backup was created. The Oracle-suggested strategy uses the incremental backup and incrementally updated backup features, providing faster recoverability than is possible when applying database changes from the archived redo log files.



Because these backups on disk are retained, you can always perform a full database recovery or a point-in-time recovery to any time within the past 24 hours, at a minimum. The recovery time could reach back as far as 48 hours. This is because just before a backup is taken on a given day, the backup from the beginning of day $n-1$ still exists.

ORACLE® Enterprise Manager Cloud Control 13c

Schedule Oracle-Suggested Backup: Review

Database: ORCL.example.com
 Backup Strategy: Oracle-Suggested Backup

Settings

Destination: Disk
 Daily Backup: A full database copy will be performed during the first backup.
 Subsequently, an incremental backup to disk will be performed every day.
 The backups on disk will be retained so that you can always perform a full database recovery or a point-in-time recovery to any time within the past day.

Fast Recovery Area: /u01/app/oracle/fast_recovery_area/ORCL

RMAN Script

The RMAN script below is generated based on previous input.

```
Daily Script:  

run {  

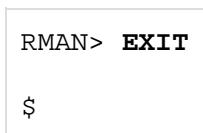
allocate channel oem_disk_backup1 type disk ;  

recover copy of database with tag 'ORA$OEM_LEVEL_0';  

backup incremental level 1 copies=1 for recover of copy with tag 'ORA$OEM_LEVEL_0' database;  

}
```

18. Exit RMAN and close the terminal window.



Practice 10-5 Creating a Partial Database Backup

Overview

In this practice, you use Recovery Manager to back up PDB1, including the archived redo log files. This is a partial database backup.

Assumptions

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

You completed the following practices:

- Practice 10-1 Configuring Your Database for Recovery
- Practice 10-3 Configuring Automatic Backups of the Control File and SPFILE
- Practice 10-4 Creating a Whole Database Backup

Tasks

Complete the following steps.

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 Recovery Manager (RMAN) and connect to the CDB root as the `SYS` user.

```
$ rman target /
...
RMAN>
```

3. Back up PDB1, including the archived redo log files.

```
RMAN> BACKUP PLUGGABLE DATABASE PDB1 PLUS ARCHIVELOG;

Starting backup at 23-JAN-17
current log archived
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=288 device type=DISK
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=47 RECID=18 STAMP=934064302
input archived log thread=1 sequence=48 RECID=21 STAMP=934064306
input archived log thread=1 sequence=49 RECID=20 STAMP=934064304
```

```
input archived log thread=1 sequence=50 RECID=17 STAMP=934064301
input archived log thread=1 sequence=51 RECID=19 STAMP=934064303
input archived log thread=1 sequence=52 RECID=15 STAMP=934064298
input archived log thread=1 sequence=53 RECID=16 STAMP=934064300
input archived log thread=1 sequence=54 RECID=24 STAMP=934064411
input archived log thread=1 sequence=55 RECID=25 STAMP=934066912
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_23/o1_mf_annnn_TAG20170123T230152_d8f2v0z0_.bkp
tag=TAG20170123T230152 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:03
Finished backup at 23-JAN-17
```

```
Starting backup at 23-JAN-17
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00050 name=/u01/app/oracle/oradata/ORCL/
pdb1/tbs_app01.dbf
input datafile file number=00043 name=/u01/app/oracle/oradata/ORCL/
pdb1/sysaux01.dbf
input datafile file number=00042 name=/u01/app/oracle/oradata/ORCL/
pdb1/system01.dbf
input datafile file number=00045 name=/u01/app/oracle/oradata/ORCL/
pdb1/users01.dbf
input datafile file number=00044 name=/u01/app/oracle/oradata/ORCL/
pdb1/undotbs01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T230156_d8f2v4cn_.bkp tag=TAG20170123T230156
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 23-JAN-17
```

```
Starting backup at 23-JAN-17
current log archived
using channel ORA_DISK_1
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=56 RECID=27 STAMP=934066923
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_23/o1_mf_annnn_TAG20170123T230203_d8f2vcn4_.bkp
tag=TAG20170123T230203 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17
```

```
Starting Control File and SPFILE Autobackup at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/
2017_01_23/o1_mf_s_934066924_d8f2vdwy_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 23-JAN-17
```

```
RMAN>
```

4. Exit RMAN.

```
RMAN> EXIT
```

```
Recovery Manager complete.  
$
```

5. Question: Did the partial backup automatically include the SPFILE and control files?

Answer: Yes. The setting completed in [Practice 10-3 Configuring Automatic Backups of the Control File and SPFILE](#) is also valid for partial backups.

6. Question: How many backup sets are created?

Answer: Four backup sets: one for the PDB data files, one for the SPFILE and control file, one for the archived log files before the data file backup set, and one for the archived log files after the data file backup set.

7. Question: Can you connect in RMAN directly to the PDB to perform the same backup?

Answer: Yes. In this case, you do not have to specify that you want to back up a PDB. Instead, you can use the simple BACKUP DATABASE command.

8. Perform a partial database backup in RMAN directly.

- Start RMAN and connect to PDB1 as the SYS user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ rman target SYS/<password>@PDB1
```

```
Recovery Manager: Release 12.2.0.1.0 - Production on Thu Mon Jan 23  
23:04:56 2017  
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights  
reserved.  
connected to target database: ORCL:PDB1 (DBID=220169531)  
RMAN>
```

- Execute the BACKUP DATABASE command. Notice that the SPFILE and control file are not backed up.

```
RMAN> BACKUP DATABASE;

Starting backup at 23-JAN-17
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=50 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00050 name=/u01/app/oracle/oradata/ORCL/pdb1/
tbs_app01.dbf
input datafile file number=00043 name=/u01/app/oracle/oradata/ORCL/pdb1/
sysaux01.dbf
input datafile file number=00042 name=/u01/app/oracle/oradata/ORCL/pdb1/
system01.dbf
input datafile file number=00045 name=/u01/app/oracle/oradata/ORCL/pdb1/
users01.dbf
input datafile file number=00044 name=/u01/app/oracle/oradata/ORCL/pdb1/
undotbs01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF494767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T230539_d8f323qd_.bkp tag=TAG20170123T230539
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 23-JAN-17
RMAN>
```

9. Try to configure the recovery setting for the PDB so that the SPFILE and control file are backed up too. You get an error message because you must be connected to the CDB root to configure any recovery settings.

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;

RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03002: failure of configure command at 01/23/2017 23:05:51
RMAN-07536: command not allowed when connected to a Pluggable Database
RMAN>
```

10. Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.
$
```

11. Back up the TBS_APP tablespace in PDB2.

- a. Connect to PDB2 as the SYS user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ rman target SYS/<password>@PDB2

Recovery Manager: Release 12.2.0.1.0 - Production on Mon Jan 23 23:07:58
2017
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights
reserved.
connected to target database: ORCL:PDB2 (DBID=572234289)
RMAN>
```

- b. Back up the tablespace.

```
RMAN> BACKUP TABLESPACE tbs_app;

Starting backup at 23-JAN-17
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=287 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00051 name=/u01/app/oracle/oradata/ORCL/pdb2/
tbs_app01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T230847_d8f38057_.bkp tag=TAG20170123T230847
comment=None
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17
RMAN>
```

- c. Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.
$
```

12. Can you connect in RMAN to the CDB root and perform the same operation?

- a. Start RMAN and connect to the CDB root as the SYS user.

```
$ rman target /
...
RMAN>
```

- b. Back up the TBS_APP tablespace in PDB2. You must specify the PDB in which the tablespace exists.

```
RMAN> BACKUP TABLESPACE PDB2:tbs_app;

Starting backup at 23-JAN-17
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=274 device type=DISK
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00051 name=/u01/app/oracle/oradata/ORCL/
pdb2/tbs_app01.dbf
channel ORA_DISK_1: starting piece 1 at 23-JAN-17
channel ORA_DISK_1: finished piece 1 at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CAD91DBF4E4767E053F511ED0A6A98/backupset/2017_01_23/
o1_mf_nnndf_TAG20170123T231022_d8f3byy1_.bkp tag=TAG20170123T231022
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 23-JAN-17

Starting Control File and SPFILE Autobackup at 23-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/
2017_01_23/o1_mf_s_934067424_d8f3c097_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 23-JAN-17
RMAN>
```

- c. Question: Did the operation back up only the tablespace data files?

Answer: No. It also backed up the SPFILE and control file. It is only when you are connected to the CDB root to perform backups that the SPFILE and control file are backed up.

- d. Exit RMAN and close the terminal window.

```
RMAN> EXIT

Recovery Manager complete.
$
```

Practice 10-6 Recovering From an Essential Data File Loss

Overview

In this practice, you recover your CDB after the data file for the SYSTEM tablespace (in the CDB root) has been inadvertently removed. You use two different commands in Recovery Manager (RMAN) to recover:

- RECOVER
- RESTORE

Assumptions

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

You completed the following practices:

- Practice 10-3 Configuring Automatic Backups of the Control File and SPFILE
- Practice 10-4 Creating a Whole Database Backup

Tasks

Create a Crash Situation

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window 1.

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

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

2. Execute the `$HOME/labs/RMAN_crash.sh` shell script to remove the data file of the SYSTEM tablespace in the CDB root.

```
$ $HOME/labs/RMAN_crash.sh

System altered.

System altered.
$
```

3. Attempt an administrative task, for example, try to create a user.

- a. Start SQL*Plus and connect to the CDB root as the `SYSTEM` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>
...
SQL>
```

- b. Create a common user named c##test. See [Appendix - Product-Specific Credentials](#) for the password. You get an error telling you that data file #1 is missing and it is the one associated with the SYSTEM tablespace in the CDB root.

```
SQL> CREATE USER c##test IDENTIFIED BY <password>;
CREATE USER c##test IDENTIFIED BY <password>
*
ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-01116: error in opening database file 1
ORA-01110: data file 1: '/u01/app/oracle/oradata/ORCL/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL>
```

- c. Exit SQL*Plus.

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

4. Consider your recovery options.

- a. Question: Which type of recovery is possible in this case?

Answer: A complete recovery is possible as long as you have all available backups required. This means that you have a backup (backup set or image copy) of the missing data file, all archive log files required to recover the restored data file up to the current SCN of the CDB including all redo log files (one member in each group will be sufficient).

- b. Question: Which methods can you use to recover the data file?

Answer: RMAN is the best utility to recover data. Either you know the RESTORE and RECOVER commands and proceed to the recovery operation, or you first get help with the LIST FAILURE command.

Perform a Recovery By Using the RESTORE Command

1. Start Recovery Manager (RMAN) and connect to the target database (CDB root).

```
$ rman target /
...
RMAN>
```

- Issue the RESTORE command. Notice that you reference the number for the missing data file. In this example, the missing data file number is 1.

```
RMAN> RESTORE DATAFILE 1;

Starting restore at 24-JAN-17
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=48 device type=DISK

channel ORA_DISK_1: restoring datafile 00001
input datafile copy RECID=5 STAMP=934064332 file name=/u01/app/oracle/
fast_recovery_area/ORCL/ORCL/datafile/o1_mf_system_d8f0b2yp_.dbf
destination for restore of datafile 00001: /u01/app/oracle/oradata/ORCL/
system01.dbf
RMAN-00571: =====
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03002: failure of restore command at 01/24/2017 00:10:01
ORA-19573: cannot obtain exclusive enqueue for datafile 1
ORA-45909: restore, recover or block media recovery may be in progress
ORA-19600: input file is datafile-copy 5 (/u01/app/oracle/fast_recovery_area/
ORCL/ORCL/datafile/o1_mf_system_d8f0b2yp_.dbf)
ORA-19601: output file is datafile 1 (/u01/app/oracle/oradata/ORCL/system01.dbf)
RMAN>
```

- Question: What does the error message "cannot obtain exclusive enqueue for datafile 1" mean?
Answer: The restore operation requires an exclusive enqueue lock on the data file 1 that is not obtainable. In this case, you have to open the database instance in MOUNT mode. This means that you have to shut down the database instance if this one did not already abort.
- Question: What does a database instance shut down imply?
Answer: This closes all PDBs and therefore prevents all users from working during the recovery operation.

5. Try shutting down the database instance in **IMMEDIATE** mode. You get an error.

```
RMAN> SHUTDOWN IMMEDIATE

RMAN-00571: =====
RMAN-00569: ====== ERROR MESSAGE STACK FOLLOWS =====
RMAN-00571: =====
RMAN-03002: failure of shutdown command at 01/24/2017 00:12:22
ORA-01116: error in opening database file 1
ORA-01110: data file 1: '/u01/app/oracle/oradata/ORCL/system01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
RMAN>
```

6. Question: Why does the **SHUTDOWN IMMEDIATE** command not work?

Answer: RMAN needs to close all data files cleanly by writing the current SCN to all data file headers. This cannot be done because data file 1 is missing.

7. Try shutting down the database instance in **ABORT** mode. This method works.

```
RMAN> SHUTDOWN ABORT

Oracle instance shut down
RMAN>
```

8. Start up the database instance in **MOUNT** mode.

```
RMAN> STARTUP MOUNT

connected to target database (not started)
Oracle instance started
database mounted

Total System Global Area     2466250752 bytes

Fixed Size                  8795760 bytes
Variable Size                671091088 bytes
Database Buffers             1778384896 bytes
Redo Buffers                 7979008 bytes
RMAN>
```

9. Restore the missing data file.

```
RMAN> RESTORE DATAFILE 1;

Starting restore at 24-JAN-17
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=255 device type=DISK

channel ORA_DISK_1: restoring datafile 00001
input datafile copy RECID=5 STAMP=934064332 file name=/u01/app/
oracle/fast_recovery_area/ORCL/ORCL/datafile/o1_mf_system_d8f0b2yp_.dbf
destination for restore of datafile 00001: /u01/app/oracle/oradata/
ORCL/system01.dbf
channel ORA_DISK_1: copied datafile copy of datafile 00001
output file name=/u01/app/oracle/oradata/ORCL/system01.dbf RECID=0 STAMP=0
Finished restore at 24-JAN-17
RMAN>
```

10. Recover the missing data file.

```
RMAN> RECOVER DATAFILE 1;

Starting recover at 24-JAN-17
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 54 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_54_d8f0dvwz_.arc archived log for thread 1 with sequence 55 is
already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_55_d8f2v01z_.arc archived log for thread 1 with sequence 56 is
already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_56_d8f2vcfq_.arc archived log for thread 1 with sequence 57 is
already on disk as file /u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_57_d8f3q9z6_.arc
archived log for thread 1 with sequence 58 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_58_d8f3qb14_.arc
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_54_d8f0dvwz_.arc thread=1 sequence=54
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_55_d8f2v01z_.arc thread=1 sequence=55
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_56_d8f2vcfq_.arc thread=1 sequence=56
media recovery complete, elapsed time: 00:00:04
Finished recover at 24-JAN-17
RMAN>
```

11. Open the CDB root.

```
RMAN> ALTER DATABASE OPEN;  
Statement processed  
RMAN>
```

12. Open all PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;  
Statement processed  
RMAN>
```

13. Exit RMAN.

```
RMAN> EXIT  
Recovery Manager complete.  
$
```

14. Start SQL*Plus and connect to the CDB root as the SYSTEM user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>  
...  
SQL>
```

15. Try creating the c##test user again. See [Appendix - Product-Specific Credentials](#) for the password. This time the user is created. You can keep window 1 open.

```
SQL> CREATE USER c##test IDENTIFIED BY <password>;  
User created.  
SQL>
```

Perform a Recovery By Using the RECOVER Command

1. Open a new terminal window and replay the crash situation. This window will be referred to as window 2.

```
$ $HOME/labs/RMAN_crash.sh  
System altered.  
System altered.  
$
```

2. In window 1, try to create another user named c##test2. See [Appendix - Product-Specific Credentials](#) for the password. The user is successfully created.

```
SQL> CREATE USER c##test2 IDENTIFIED BY <password>;  
User created.  
SQL>
```

3. Question: How is it possible that you were able to create the user immediately following the crash scenario?
Answer: Remember that the DBWR background process does not necessarily write immediately into the data files.
4. In window 1, attempt another operation like the following one. If it completes too, try the ALTER SYSTEM SWITCH LOGFILE command. You should receive an error message about the missing data file.

```
SQL> ALTER DATABASE DATAFILE 1 RESIZE 1G;  
  
ALTER DATABASE DATAFILE 1 RESIZE 1G  
*  
ERROR at line 1:  
ORA-01565: error in identifying file  
' /u01/app/oracle/oradata/ORCL/system01.dbf '  
ORA-27037: unable to obtain file status  
Linux-x86_64 Error: 2: No such file or directory  
Additional information: 7  
  
SQL>
```

5. In window 1, exit SQL*Plus.

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

6. In window 1, start RMAN and connect to the target database. If you get the following error message telling you that RMAN cannot find the system01.dbf file, proceed with substeps a through c. If you did not get any error message, then continue on to step 7.

```
$ rman target /  
  
Recovery Manager: Release 12.2.0.1.0 - Production on Tue Jan 24 00:20:37 2017  
  
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.  
  
RMAN-00571: =====  
RMAN-00569: ===== ERROR MESSAGE STACK FOLLOWS =====  
RMAN-00571: =====  
RMAN-00554: initialization of internal recovery manager package failed  
RMAN-06003: ORACLE error from target database:  
ORA-00604: error occurred at recursive SQL level 3  
ORA-01116: error in opening database file 1  
ORA-01110: data file 1: '/u01/app/oracle/oradata/ORCL/system01.dbf'  
ORA-27041: unable to open file  
Linux-x86_64 Error: 2: No such file or directory  
Additional information: 3  
$
```

- a. In window 1, start SQL*Plus and connect to the CDB root as the SYS user with the SYSDBA privilege.

```
$ sqlplus / AS SYSDBA  
...  
SQL>
```

- b. Shut down the database instance in ABORT mode.

```
SQL> SHUTDOWN ABORT  
  
ORACLE instance shut down.  
SQL>
```

- c. Exit SQL*Plus.

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

- d. In window 1, start RMAN and connect to the target database. A message states that the target database is not started.

```
$ rman target /  
  
Recovery Manager: Release 12.2.0.1.0 - Production on Mon Jan 24 00:20:40  
2017  
  
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights  
reserved.  
  
connected to target database (not started)  
RMAN>
```

7. Start up the database instance in MOUNT mode.

```
RMAN> STARTUP MOUNT;  
  
Oracle instance started  
database mounted  
  
Total System Global Area      2466250752 bytes  
  
Fixed Size                  8795760 bytes  
Variable Size                671091088 bytes  
Database Buffers             1778384896 bytes  
Redo Buffers                 7979008 bytes  
RMAN>
```

8. Get help with the LIST FAILURE command. The Summary column tells you that system01.dbf is missing.

```
RMAN> LIST FAILURE;

using target database control file instead of recovery catalog
Database Role: PRIMARY

List of Database Failures
=====
Failure ID Priority Status      Time Detected Summary
-----  -----  -----  -----
822      CRITICAL OPEN       23-JAN-17    System datafile 1: '/u01/app/oracle/
oradata/ORCL/system01.dbf' is missing
RMAN>
```

9. Display repair options. At the very end of the results, a repair script is listed. In this example, the script is /u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_3395366943.hm . Your file name will be slightly different.

```
RMAN> ADVISE FAILURE;

Database Role: PRIMARY

List of Database Failures
=====

Failure ID Priority Status      Time Detected Summary
----- ----- -----
822       CRITICAL OPEN        23-JAN-17      System datafile 1: '/u01/app/
oracle/oradata/ORCL/system01.dbf' is missing

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=258 device type=DISK
analyzing automatic repair options complete

Mandatory Manual Actions
=====
no manual actions available

Optional Manual Actions
=====
1. If file /u01/app/oracle/oradata/ORCL/system01.dbf was unintentionally
renamed or moved, restore it

Automated Repair Options
=====
Option Repair Description
-----
1     Restore and recover datafile 1
      Strategy: The repair includes complete media recovery with no data loss
      Repair script: /u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_3496690698.hm

RMAN>
```

10. Do a repair failure preview. This action does not make any repairs. Instead, it generates a script with all repair actions and comments.

```
RMAN> REPAIR FAILURE PREVIEW;

Strategy: The repair includes complete media recovery with no data loss
Repair script: /u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_3395366943.hm

contents of repair script:
# restore and recover datafile
restore ( datafile 1 );
recover datafile 1;
sql 'alter database datafile 1 online';
RMAN>
```

11. Use the REPAIR FAILURE command to repair database failures identified by the Data Recovery Advisor. When prompted, enter YES to execute the repair. When prompted to open the database, enter YES. If you get an error, try running the REPAIR FAILURE command again.

```
RMAN> REPAIR FAILURE;

Strategy: The repair includes complete media recovery with no data loss
Repair script: /u01/app/oracle/diag/rdbms/orcl/ORCL/hm/reco_3496690698.hm

contents of repair script:
# restore and recover datafile
restore ( datafile 1 );
recover datafile 1;
sql 'alter database datafile 1 online';

Do you really want to execute the above repair (enter YES or NO)? YES
executing repair script

Starting restore at 24-JAN-17
using channel ORA_DISK_1

channel ORA_DISK_1: restoring datafile 00001
input datafile copy RECID=5 STAMP=934064332 file name=/u01/app/oracle/
fast_recovery_area/ORCL/ORCL/datafile/o1_mf_system_d8f0b2yp_.dbf
destination for restore of datafile 00001: /u01/app/oracle/oradata/
ORCL/system01.dbf
channel ORA_DISK_1: copied datafile copy of datafile 00001
output file name=/u01/app/oracle/oradata/ORCL/system01.dbf RECID=0
STAMP=0
Finished restore at 24-JAN-17

Starting recover at 24-JAN-17
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 54 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_54_d8f0dvwz_.arc
archived log for thread 1 with sequence 55 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_55_d8f2v01z_.arc
archived log for thread 1 with sequence 56 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_56_d8f2vcfq_.arc
archived log for thread 1 with sequence 57 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_57_d8f3q9z6_.arc
archived log for thread 1 with sequence 58 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_23/
o1_mf_1_58_d8f3qb14_.arc
archived log for thread 1 with sequence 59 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_59_d8f7bfvo_.arc
```

```
archived log for thread 1 with sequence 60 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_60_d8f7dft8_.arc
archived log for thread 1 with sequence 61 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24
o1_mf_1_61_d8f7dfvf_.arc
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_54_d8f0dvwz_.arc thread=1 sequence=54
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_55_d8f2v01z_.arc thread=1 sequence=55
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_56_d8f2vcfq_.arc thread=1 sequence=56
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_57_d8f3q9z6_.arc thread=1 sequence=57
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_23/o1_mf_1_58_d8f3qb14_.arc thread=1 sequence=58
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_24/o1_mf_1_59_d8f7bfvo_.arc thread=1 sequence=59
media recovery complete, elapsed time: 00:00:04
Finished recover at 24-JAN-17
```

```
sql statement: alter database datafile 1 online
repair failure complete
```

```
Do you want to open the database (enter YES or NO)? YES
```

```
database opened  
RMAN>
```

12. Open all the PDBs.

```
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;  
  
Statement processed  
  
RMAN>
```

13. Exit RMAN.

```
RMAN> EXIT  
  
Recovery Manager complete.  
$
```

14. In window 1, attempt to resize the data file again.

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

```
$ sqlplus / AS SYSDBA  
  
...  
SQL>
```

- Try to resize the data file. The resize is successful, indicating that your database has recovered.

```
SQL> ALTER DATABASE DATAFILE 1 RESIZE 1G;  
  
Database altered.  
SQL>
```

- Exit SQL*Plus and close both terminal windows (window 1 and window 2).

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

Practice 10-7 Recovering From an Application Data File Loss

Overview

In this practice, you recover the loss of a PDB data file that has been inadvertently removed. To recover, you use two different methods in Recovery Manager (RMAN):

- RESTORE and RECOVER
- REPAIR (new command in Oracle Database 12.2)

Assumptions

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

You completed practice [Practice 10-1 Configuring Your Database for Recovery](#).

Tasks

Set Up Your Environment

In this section, you recreate `PDB1` and back it up.

1. Open a new terminal window and execute the `$HOME/labs/setup_pdb1.sh` shell script. This script recreates `PDB1` with a `TBS_APP` tablespace and `OE` schema.

```
$ $HOME/labs/setup_pdb1.sh  
...  
$
```

2. 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 RMAN and connect to the CDB root as the `SYS` user.

```
$ rman target /  
...  
RMAN>
```

4. Back up `PDB1`, including the archive logs.

```
RMAN> BACKUP PLUGGABLE DATABASE pdb1 PLUS ARCHIVELOG;  
Starting backup at 24-JAN-17  
current log archived
```

```
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=273 device type=DISK
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=47 RECID=18 STAMP=934064302
input archived log thread=1 sequence=48 RECID=21 STAMP=934064306
input archived log thread=1 sequence=49 RECID=20 STAMP=934064304
input archived log thread=1 sequence=50 RECID=17 STAMP=934064301
input archived log thread=1 sequence=51 RECID=19 STAMP=934064303
input archived log thread=1 sequence=52 RECID=15 STAMP=934064298
input archived log thread=1 sequence=53 RECID=16 STAMP=934064300
input archived log thread=1 sequence=54 RECID=24 STAMP=934064411
input archived log thread=1 sequence=55 RECID=25 STAMP=934066912
input archived log thread=1 sequence=56 RECID=27 STAMP=934066923
input archived log thread=1 sequence=57 RECID=31 STAMP=934067818
input archived log thread=1 sequence=58 RECID=29 STAMP=934067818
input archived log thread=1 sequence=59 RECID=33 STAMP=934071502
input archived log thread=1 sequence=60 RECID=35 STAMP=934071565
input archived log thread=1 sequence=61 RECID=37 STAMP=934071565
input archived log thread=1 sequence=62 RECID=39 STAMP=934072232
input archived log thread=1 sequence=63 RECID=41 STAMP=934073230
channel ORA_DISK_1: starting piece 1 at 24-JAN-17
channel ORA_DISK_1: finished piece 1 at 24-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_24/o1_mf_annnn_TAG20170124T004710_d8f90h69_.bkp
tag=TAG20170124T004710 comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 24-JAN-17
```

```
Starting backup at 24-JAN-17
using channel ORA_DISK_1
channel ORA_DISK_1: starting full datafile backup set
channel ORA_DISK_1: specifying datafile(s) in backup set
input datafile file number=00056 name=/u01/app/oracle/oradata/ORCL/pdb1/
tbs_app01.dbf
input datafile file number=00053 name=/u01/app/oracle/oradata/ORCL/pdb1/
sysaux01.dbf
input datafile file number=00052 name=/u01/app/oracle/oradata/ORCL/pdb1/
system01.dbf
input datafile file number=00055 name=/u01/app/oracle/oradata/ORCL/pdb1/
users01.dbf
input datafile file number=00054 name=/u01/app/oracle/oradata/ORCL/pdb1/
undotbs01.dbf
channel ORA_DISK_1: starting piece 1 at 24-JAN-17
channel ORA_DISK_1: finished piece 1 at 24-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
46CD713E045C58ABE053F511ED0A5743/backupset/2017_01_24/
o1_mf_nnndf_TAG20170124T004718_d8f90ps6_.bkp tag=TAG20170124T004718
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:07
Finished backup at 24-JAN-17
```

```
Starting backup at 24-JAN-17
current log archived
using channel ORA_DISK_1
```

```
channel ORA_DISK_1: starting archived log backup set
channel ORA_DISK_1: specifying archived log(s) in backup set
input archived log thread=1 sequence=64 RECID=43 STAMP=934073245
channel ORA_DISK_1: starting piece 1 at 24-JAN-17
channel ORA_DISK_1: finished piece 1 at 24-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/backupset/
2017_01_24/o1_mf_annnn_TAG20170124T004725_d8f90xvc_.bkp tag=TAG20170124T004725
comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:01
Finished backup at 24-JAN-17

Starting Control File and SPFILE Autobackup at 24-JAN-17
piece handle=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/autobackup/2017_01_24/
o1_mf_s_934073247_d8f90z84_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 24-JAN-17
RMAN>
```

5. Exit RMAN.

```
RMAN> EXIT

Recover Manager complete.
$
```

Remove a Data File

In this section, you run a script that removes a data file from PDB1. You research the problem and discover which data file is missing.

1. Execute the following script.

```
$ $HOME/labs/RMAN_crash_app.sh

System altered.

System altered.
$
```

2. Perform an administrative task, for example, create an application table and insert data into it.

a. Connect to the CDB root as the SYS user with the SYSDBA privilege.

```
$ sqlplus / AS SYSDBA

...
SQL>
```

b. Connect to PDB1 as the SYS user.

```
SQL> ALTER SESSION SET CONTAINER = pdb1;  
Session altered.  
SQL>
```

- c. Create a table named OE . TEST. The results show that the table is created.

```
SQL> CREATE TABLE oe.test (c NUMBER);  
Table created.  
SQL>
```

- d. Try to insert rows into the table. You get an error indicating that the data file named tbs_app01.dbf is missing in PDB1. Note that in this example the data file number that represents tbs_app01.dbf is 56. Identify and remember the file number assigned to tbs_app01.dbf on your own system. Your number will probably differ from this example.

```
SQL> INSERT INTO oe.test VALUES (1);  
insert into oe.test values (1)  
*  
ERROR at line 1:  
ORA-01116: error in opening database file 56  
ORA-01110: data file 56:  
'/u01/app/oracle/oradata/ORCL/pdb1/tbs_app01.dbf'  
ORA-27041: unable to open file  
Linux-x86_64 Error: 2: No such file or directory  
Additional information: 3  
SQL>
```

3. Question: Why does it fail during the INSERT statement and not during the CREATE TABLE statement? Note: It could also have failed during the CREATE TABLE statement.

Answer: When you create a table, the segment is not created yet. The CREATE statement updates a table in the SYSTEM tablespace. Only when the first row is inserted into the table is the segment created in the data file.

4. Find out the tablespace to which tbs_app01.dbf belongs. In this command, substitute the number assigned to tbs_app01.dbf on your own system, that you identified in Step 2d, for the example value 56. The result shows that the tablespace is TBS_APP.

```
SQL> SELECT tablespace_name from dba_data_files WHERE file_id = 56;  
TABLESPACE_NAME  
-----  
TBS_APP  
SQL>
```

5. Check if the TBS_APP tablespace is still online. The results indicate that it is online, and so are all the other tablespaces in PDB1.

```
SQL> SELECT tablespace_name, status FROM dba_tablespaces;
TABLESPACE_NAME    STATUS
-----
SYSTEM            ONLINE
SYSAUX            ONLINE
UNDOTBS1          ONLINE
TEMP              ONLINE
USERS             ONLINE
TBS_APP           ONLINE
6 rows selected.
SQL>
```

6. Check if PDB1 is still open. The results indicate that PDB1 is still open.

```
SQL> SHOW PDBS

  CON_ID CON_NAME          OPEN MODE RESTRICTED
----- -----
      3  PDB1            READ WRITE NO
SQL>
```

7. Exit SQL*Plus.

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

Restore and Recover PDB1

In this section, you restore and recover PDB1 in Recovery Manager.

1. Question: Which type of recovery is possible in this case?
Answer: A complete recovery is possible as long as you have all available backups required. This means that you have a backup (backup set or image copy) of the missing data file and all archive log files required to recover the restored data file up to the current SCN of the PDB including all redo log files (one member in each group will be sufficient).
2. Question: Which methods can you use to recover?
Answer: RMAN is the best utility to recover data. Either you know the RESTORE and RECOVER commands and proceed to the recovery operation, or you get help with the LIST FAILURE commands. You can also use the new 12.2 simple REPAIR command.
3. Start RMAN and connect to PDB1 as the SYS user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ rman target sys/<password>@PDB1
Recovery Manager: Release 12.2.0.1.0 - Production on Tue Jan 24 00:54:55 2017
Copyright (c) 1982, 2016, Oracle and/or its affiliates. All rights reserved.
connected to target database: ORCL:PDB1 (DBID=2929991525)
RMAN>
```

4. Issue the `REPORT SCHEMA` command to list the names of the data files (permanent and temporary) and tablespaces for PDB1. The example data file number 56 is part of the TBS_APP tablespace in PDB1 and shows a size of 0MB. Identify the entry for the TBS_APP tablespace on your system.

```
RMAN> REPORT SCHEMA;

using target database control file instead of recovery catalog
Report of database schema for database with db_unique_name ORCL

List of Permanent Datafiles
=====
File Size(MB) Tablespace          RB segs Datafile Name
----- ----- -----
52   250    SYSTEM               NO      /u01/app/oracle/oradata/ORCL/
pdb1/system01.dbf
53   360    SYSAUX              NO      /u01/app/oracle/oradata/ORCL/
pdb1/sysaux01.dbf
54   100    UNDOTBS1            NO      /u01/app/oracle/oradata/ORCL/
pdb1/undotbs01.dbf
55   250    USERS                NO      /u01/app/oracle/oradata/ORCL/
pdb1/users01.dbf
56   0     TBS_APP              NO      /u01/app/oracle/oradata/ORCL/
pdb1/tbs_app01.dbf

List of Temporary Files
=====
File Size(MB) Tablespace          Maxsize(MB) Tempfile Name
----- ----- -----
3     64     TEMP                32767    /u01/app/oracle/oradata/ORCL/
pdb1/temp012016-12-20_01-33-16-961-AM.dbf

RMAN>
```

5. You must close PDB1 in `IMMEDIATE` mode (which puts the PDB into `MOUNTED` mode) before restoring the PDB; otherwise, you will get the error "cannot obtain exclusive enqueue for datafile...".

```
RMAN> SHUTDOWN IMMEDIATE;

database closed
RMAN>
```

6. Restore PDB1.

```
RMAN> RESTORE DATABASE;

Starting restore at 24-JAN-17
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=274 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00052 to /u01/app/oracle/oradata/
ORCL/pdb1/system01.dbf
channel ORA_DISK_1: restoring datafile 00053 to /u01/app/oracle/oradata/
ORCL/pdb1/sysaux01.dbf
channel ORA_DISK_1: restoring datafile 00054 to /u01/app/oracle/oradata/
ORCL/pdb1/undotbs01.dbf
channel ORA_DISK_1: restoring datafile 00055 to /u01/app/oracle/oradata/
ORCL/pdb1/users01.dbf
channel ORA_DISK_1: restoring datafile 00056 to /u01/app/oracle/oradata/
ORCL/pdb1/tbs_app01.dbf
channel ORA_DISK_1: reading from backup piece /u01/app/oracle/
fast_recovery_area/ORCL/ORCL/46CD713E045C58ABE053F511ED0A5743/backupset/
2017_01_24/o1_mf_nnndf_TAG20170124T004718_d8f90ps6_.bkp
channel ORA_DISK_1: piece handle=/u01/app/oracle/fast_recovery_area/ORCL/
ORCL/46CD713E045C58ABE053F511ED0A5743/backupset/2017_01_24/
o1_mf_nnndf_TAG20170124T004718_d8f90ps6_.bkp tag=TAG20170124T004718
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:16
Finished restore at 24-JAN-17

RMAN>
```

7. Recover the database. Notice the line "starting media recovery." If you had tried to restore and recover just the data file or just the tablespace, you might have encountered media recovery errors for other data files in the tablespace.

```
RMAN> RECOVER DATABASE;

Starting recover at 24-JAN-17
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 64 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_64_d8f90xot_.arc
archived log for thread 1 with sequence 65 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_65_d8f96v49_.arc
archived log for thread 1 with sequence 66 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_66_d8f96y5f_.arc
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_24/o1_mf_1_64_d8f90xot_.arc thread=1 sequence=64
media recovery complete, elapsed time: 00:00:00
Finished recover at 24-JAN-17

RMAN>
```

8. Start up PDB1.

```
RMAN> STARTUP

database opened
RMAN>
```

9. Issue the REPORT SCHEMA command again. Notice that the TBS_APP tablespace now has a size of 800MB.

```
RMAN> REPORT SCHEMA;

Report of database schema for database with db_unique_name ORCL

List of Permanent Datafiles
=====
File Size(MB) Tablespace          RB segs Datafile Name
----- ----- -----
52    250      SYSTEM            NO      /u01/app/oracle/oradata/
ORCL/pdb1/system01.dbf
53    360      SYSAUX           NO      /u01/app/oracle/oradata/
ORCL/pdb1/sysaux01.dbf
54    100      UNDOTBS1         NO      /u01/app/oracle/oradata/
ORCL/pdb1/undotbs01.dbf
55    250      USERS             NO      /u01/app/oracle/oradata/
ORCL/pdb1/users01.dbf
56    800      TBS_APP          NO      /u01/app/oracle/oradata/
ORCL/pdb1/tbs_app01.dbf

List of Temporary Files
=====
File Size(MB) Tablespace          Maxsize(MB) Tempfile Name
----- ----- -----
3     64       TEMP             32767   /u01/app/oracle/oradata/ORCL/
pdb1/temp012016-12-20_01-33-16-961-AM.dbf

RMAN>
```

10. Exit RMAN.

```
RMAN> EXIT

Recovery Manager complete.
$
```

11. Try to insert data again into the OE.TEST table in PDB1.

- Connect to the CDB root as the SYS user with the SYSDBA privilege.

```
$ sqlplus / AS SYSDBA

...
SQL>
```

- Connect to PDB1 as the SYS user.

```
SQL> ALTER SESSION SET CONTAINER = pdb1;

Session altered.
SQL>
```

- Try to insert rows into the table. You can insert data, which means the data file is repaired.

```
SQL> INSERT INTO oe.test VALUES (1);

1 row created.

SQL>
```

12. Exit SQL*Plus.

```
SQL> EXIT

...
$
```

Use the REPAIR Command

In this section, you recreate the environment where you have a missing data file. You then use RMAN's REPAIR command to perform all the necessary operations (for example, restore and recovery) to fully recover the data file.

1. Execute the following script to remove the data file named tbs_app01.dbf from PDB1 again. In this example, the script removes data file #56. Your data file number will most likely be different.

```
$ $HOME/labs/RMAN_crash_app.sh

System altered.

System altered.

$
```

2. Perform an administrative task, for example, create an application table and try to insert data into it.

- a. Connect to the CDB root as the SYS user with the SYSDBA privilege.

```
$ sqlplus / AS SYSDBA

...
SQL>
```

- b. Connect to PDB1 as the SYS user.

```
SQL> ALTER SESSION SET CONTAINER = pdb1;

Session altered.

SQL>
```

- c. Show the open mode of PDB1. The result is showing that PDB1 is still open.

SQL> SHOW PDBS		OPEN	MODE	RESTRICTED
CON_ID	CON_NAME			
3	PDB1	READ	WRITE	NO

- d. Create a table named OE.TEST2. The result shows that the table is created.

```
SQL> CREATE TABLE oe.test2 (c NUMBER);

Table created.
SQL>
```

- e. Try to insert rows into the table. You get the same error that you did before: unable to open file. Again, you are not able to insert data into the OE schema because the data file is missing. Note that the database file number has not changed; 56 in this example.

```
SQL> INSERT INTO oe.test2 VALUES (1);

insert into oe.test2 values (1)
*
ERROR at line 1:
ORA-01116: error in opening database file 56
ORA-01110: data file 56:
'/u01/app/oracle/oradata/ORCL/pdb1/tbs_app01.dbf'
ORA-27041: unable to open file
Linux-x86_64 Error: 2: No such file or directory
Additional information: 3
SQL>
```

- f. Close PDB1 to put the PDB into MOUNTED mode.

Important: You do not want to end this SQL*Plus session without closing PDB1 first; otherwise, you may get errors when performing the REPAIR operation in RMAN.

```
SQL> ALTER PLUGGABLE DATABASE pdb1 CLOSE;

Pluggable database altered.
SQL>
```

3. Exit SQL*Plus.

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

4. Start RMAN and log in to the target database as the SYS user.

```
$ rman target /
...
RMAN>
```

5. Start a single REPAIR command. This command restores and recovers the data file in one step.

```
RMAN> REPAIR PLUGGABLE DATABASE pdb1;

Starting restore at 24-JAN-17
using target database control file instead of recovery catalog
```

```
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=17 device type=DISK
Executing: alter database datafile 52 offline
Executing: alter database datafile 53 offline
Executing: alter database datafile 54 offline
Executing: alter database datafile 55 offline
Executing: alter database datafile 56 offline

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00052 to /u01/app/oracle/oradata/
ORCL/pdb1/system01.dbf
channel ORA_DISK_1: restoring datafile 00053 to /u01/app/oracle/oradata/
ORCL/pdb1/sysaux01.dbf
channel ORA_DISK_1: restoring datafile 00054 to /u01/app/oracle/oradata/
ORCL/pdb1/undotbs01.dbf
channel ORA_DISK_1: restoring datafile 00055 to /u01/app/oracle/oradata/
ORCL/pdb1/users01.dbf
channel ORA_DISK_1: restoring datafile 00056 to /u01/app/oracle/oradata/
ORCL/pdb1/tbs_app01.dbf
channel ORA_DISK_1: reading from backup piece /u01/app/oracle/
fast_recovery_area/ORCL/ORCL/46CD713E045C58ABE053F511ED0A5743/backupset/
2017_01_24/o1_mf_nnndf_TAG20170124T004718_d8f90ps6_.bkp
channel ORA_DISK_1: piece handle=/u01/app/oracle/fast_recovery_area/ORCL/
ORCL/46CD713E045C58ABE053F511ED0A5743/backupset/2017_01_24/
o1_mf_nnndf_TAG20170124T004718_d8f90ps6_.bkp tag=TAG20170124T004718
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:18
Finished restore at 24-JAN-17

Starting recover at 24-JAN-17
using channel ORA_DISK_1

starting media recovery

archived log for thread 1 with sequence 64 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/
2017_01_24/o1_mf_1_64_d8f90xot_.arc
archived log for thread 1 with sequence 65 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_65_d8f96v49_.arc
archived log for thread 1 with sequence 66 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_66_d8f96y5f_.arc
archived log for thread 1 with sequence 67 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_67_d8fb707o_.arc
archived log for thread 1 with sequence 68 is already on disk as file
/u01/app/oracle/fast_recovery_area/ORCL/ORCL/archivelog/2017_01_24/
o1_mf_1_68_d8fb709n_.arc
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_24/o1_mf_1_64_d8f90xot_.arc thread=1 sequence=64
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_24/o1_mf_1_65_d8f96v49_.arc thread=1 sequence=65
archived log file name=/u01/app/oracle/fast_recovery_area/ORCL/ORCL/
archivelog/2017_01_24/o1_mf_1_66_d8f96y5f_.arc thread=1 sequence=66
```

```
media recovery complete, elapsed time: 00:00:00
Executing: alter database datafile 52 online
Executing: alter database datafile 53 online
Executing: alter database datafile 54 online
Executing: alter database datafile 55 online
Executing: alter database datafile 56 online
Finished recover at 24-JAN-17
```

```
RMAN>
```

6. Open PDB1.

```
RMAN> ALTER PLUGGABLE DATABASE pdb1 OPEN;
Statement processed
RMAN>
```

7. Exit RMAN.

```
RMAN> EXIT
...
$
```

8. Try to insert data into the OE.TEST2 table again.

- Start SQL*Plus and connect to PDB1 as the SYSTEM user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus SYSTEM/<password>@PDB1
...
SQL>
```

- Issue the INSERT command. The operation succeeds, which means the REPAIR command recovered the data file.

```
SQL> INSERT INTO oe.test2 VALUES (2);
1 row created.
SQL>
```

- Commit the transactions.

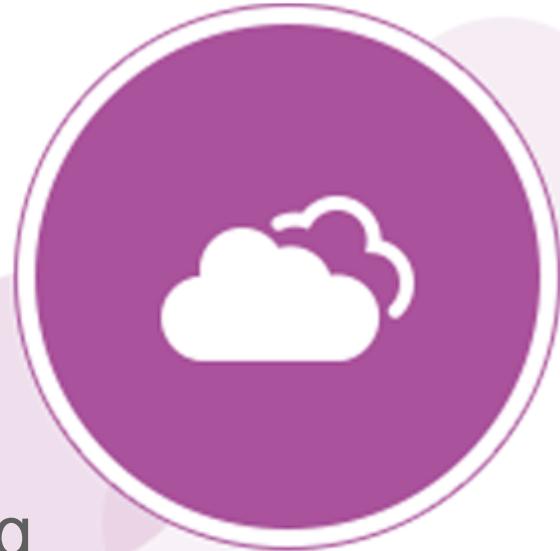
```
SQL> COMMIT;
Commit complete.
SQL>
```

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

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

11

Monitoring and Tuning Database Performance



ORACLE®

Objectives for Lesson 11

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

After completing this lesson, you should be able to:

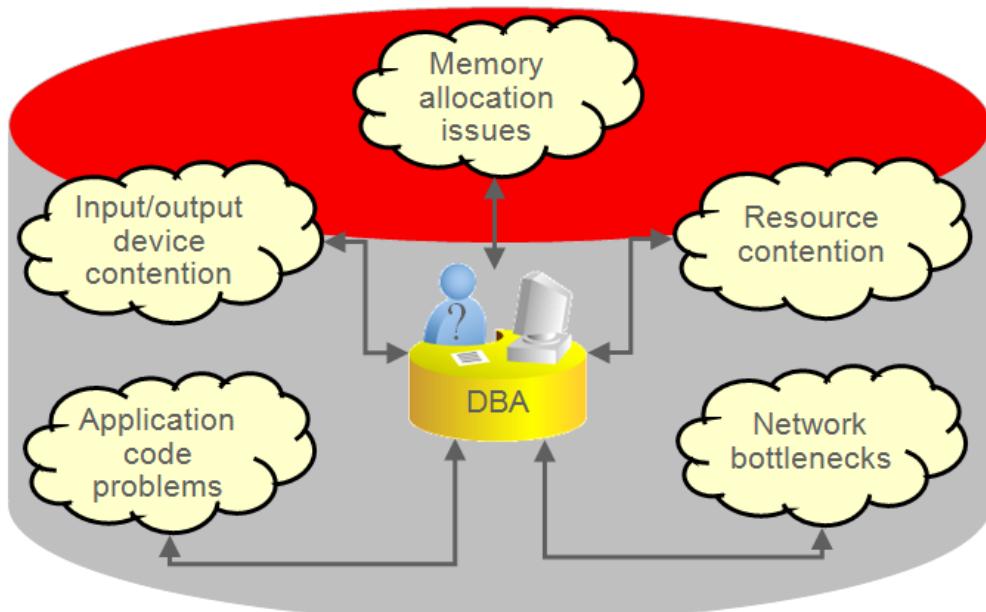
- Describe the activities that you perform to manage database performance
- Use Enterprise Manager Database Express and performance views to monitor database instance performance
- Describe the Oracle performance tuning methodology
- Describe the server statistics and metrics that are collected by the Oracle Database server
- Configure and monitor memory components for optimal performance



ORACLE®

Performance Management Activities

THESE eKIT MATERIALS ARE FOR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



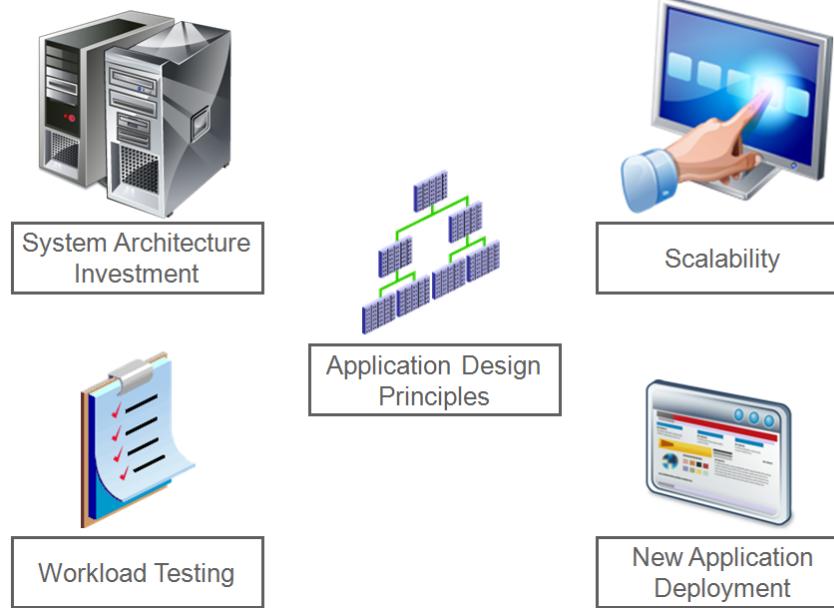
ORACLE®

Performance management includes the following activities:

- **Performance planning** is the process of establishing the environment: the hardware, software, operating system, network infrastructure, and so on.
- **Performance monitoring** is an activity that helps the DBA to locate bottlenecks and correct problem areas.
- **Instance tuning** is the actual adjustment of Oracle Database parameters and operating system (OS) parameters to gain better performance of the Oracle Database.
- **SQL tuning** involves making your application submit efficient SQL statements. SQL tuning is performed for the application as a whole, as well as for individual statements. At the application level, you want to be sure that different parts of the application are taking advantage of each other's work and are not competing for resources unnecessarily.

A DBA can look at hundreds of performance measurements, covering everything from network performance and disk input/output (I/O) speed to the time spent working on individual application operations. These performance measurements are commonly referred to as *database metrics*.

Performance Planning Considerations



ORACLE®

There are many facets to performance planning. Planning must include a balance between performance (speed), cost, and reliability.

Investment in System Architecture: You must consider the investment in your system architecture—the hardware and software infrastructure needed to meet your requirements. This, of course, requires analysis to determine the value for your given environment, application, and performance requirements. For example, the number of hard drives and controllers has an impact on the speed of data access.

Scalability: The ability of an application to scale is also important. This means that you are able to handle more and more users, clients, sessions, or transactions, without incurring a huge impact on overall system performance. The most obvious violator of scalability is serializing operations among users. If all users go through a single path one at a time, then, as more users are added, there are definitely adverse effects on performance. This is because more and more users line up to go through that path. Poorly written SQL also affects scalability. It requires many users to wait for inefficient SQL to complete; each user competing with the other on a large number of resources that they are not actually in need of.

Application Design Principles: The principles of application design can greatly affect performance. Simplicity of design, use of views and indexes, and data modeling are all very important.

Workload Testing: Any application must be tested under a representative production workload. This requires estimating database size and workload, and generating test data and system load.

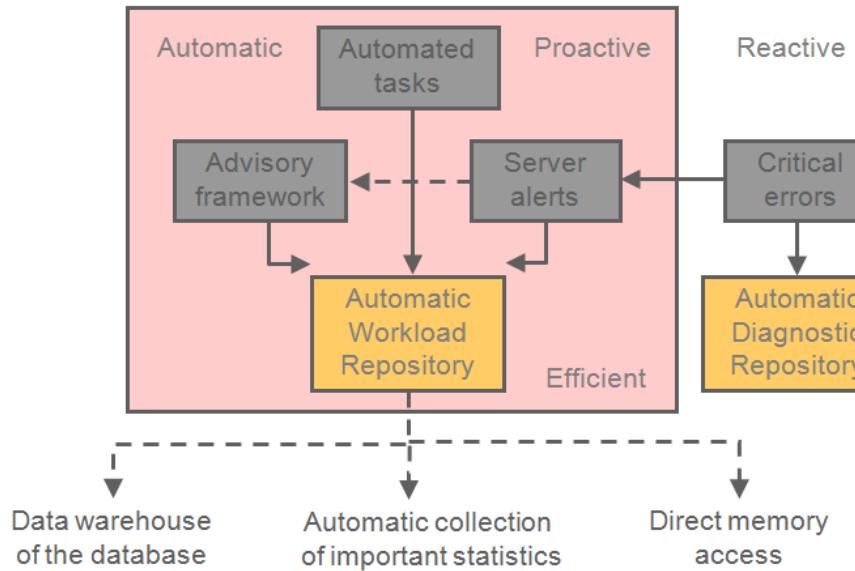
Deployment of New Applications: Performance must be considered as new applications (or new versions of applications) are deployed. Sometimes, design decisions are made to maintain compatibility with old systems during

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

the rollout. A new database should be configured (on the basis of the production environment) specifically for the applications that it hosts.

A difficult and necessary task is testing the existing applications when changing the infrastructure—for example, upgrading the database to a newer version, or changing the operating system or server hardware. Before the application is deployed for production in the new configuration, you want to know the impact. The application will almost certainly require additional tuning. You need to know that the critical functionality will perform, without errors.

Database Maintenance



ORACLE®

Proactive database maintenance is made easy by the sophisticated infrastructure of the Oracle Database, including the following main elements:

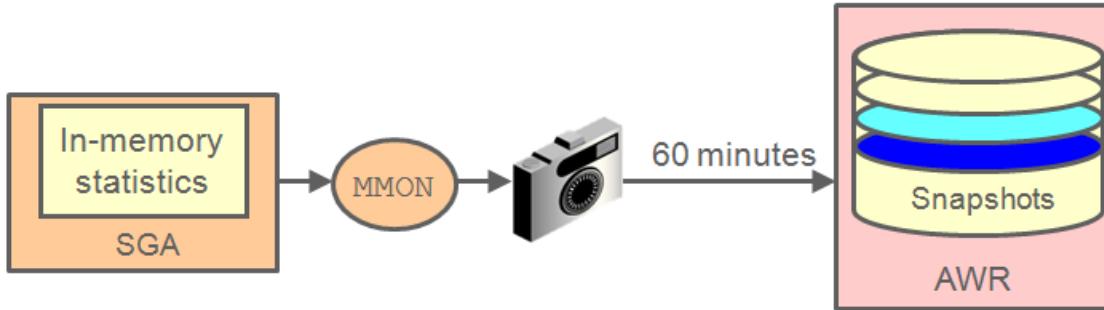
- The **Automatic Workload Repository (AWR)** is a built-in repository in each Oracle database. At regular intervals, the Oracle Database server makes a snapshot of all its vital statistics and workload information and stores this data in the AWR. The captured data can be analyzed by you, by the database server itself, or by both.
- Using automated tasks, the database server performs routine maintenance operations, such as regular backups, refreshing optimizer statistics, and database health checks.

Reactive database maintenance includes critical errors and conditions discovered by database health checkers:

- For problems that cannot be resolved automatically and require administrators to be notified (such as running out of space), the Oracle Database server provides server-generated alerts. The Oracle Database server, by default, monitors itself and sends out alerts to notify you of problems. The alerts notify you and often also provide recommendations on how to resolve the reported problem. The DBA can also be alerted by users whose transactions are locked by other users' transactions and are waiting for locks to be released.
- Recommendations are generated from several advisors, each of which is responsible for a subsystem. For example, there are memory, segment, and SQL advisors.

Automatic Workload Repository

- Built-in repository of performance information
- Snapshots of database metrics taken every 60 minutes and retained for eight days
- Foundation for all self-management functions



ORACLE

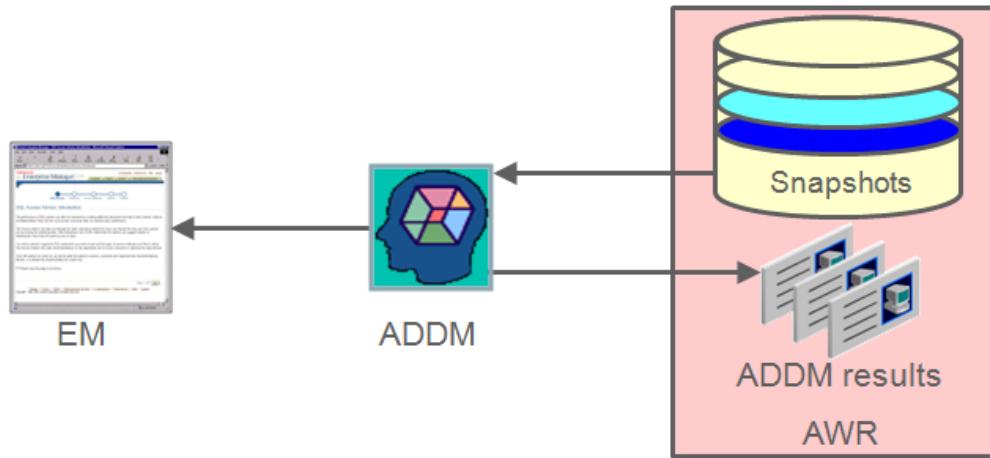
The Automatic Workload Repository (AWR) is the infrastructure that provides services to Oracle Database components to collect, maintain, and use statistics for problem detection and self-tuning purposes. You can view it as a data warehouse for database statistics, metrics, and so on.

Every 60 minutes (by default), the database automatically captures statistical information from the SGA and stores it in the AWR in the form of snapshots. These snapshots are stored on disk by a background process called Manageability Monitor (MMON). By default, snapshots are retained for eight days. You can modify both the snapshot interval and the retention intervals.

The AWR contains hundreds of tables, all belonging to the `SYS` schema and stored in the `SYSAUX` tablespace. Oracle recommends that the repository be accessed only through Enterprise Manager or the `DBMS_WORKLOAD_REPOSITORY` package to work with the AWR. Direct data manipulation language (DML) commands against the repository tables are not supported.

Automatic Database Diagnostic Monitor

- Runs after each AWR snapshot
- Monitors the instance; detects bottlenecks
- Stores results in the AWR



ORACLE®

Unlike the other advisors, the ADDM runs automatically after each AWR snapshot. Each time a snapshot is taken, the ADDM performs an analysis of the period corresponding to the last two snapshots. The ADDM proactively monitors the instance and detects most bottlenecks before they become a significant problem.

In many cases, the ADDM recommends solutions for detected problems and even quantifies the benefits for the recommendations.

Some common problems that are detected by the ADDM:

- CPU bottlenecks
- Poor Oracle Net connection management
- Lock contention
- Input/output (I/O) capacity
- Undersized database instance memory structures
- High-load SQL statements

The results of each ADDM analysis are stored in the AWR and are also accessible through Enterprise Manager.

Performance Monitoring

Instance/Database

V\$DATABASE
V\$ INSTANCE
V\$ PARAMETER
V\$ SPPARAMETER
V\$ SYSTEM_PARAMETER
V\$ PROCESS
V\$BGPROCESS
V\$PX_PROCESS_SYSSTAT
V\$SYSTEM_EVENT

Disk

V\$DATAFILE
V\$FILESTAT
V\$LOG
V\$LOG_HISTORY
V\$DBFILE
V\$TEMPFILE
V\$TEMPSEG_USAGE
V\$SEGMENT_STATISTICS

Memory

V\$BUFFER_POOL_STATISTICS
V\$LIBRARYCACHE
V\$SGAINFO
V\$PGASTAT

Contention

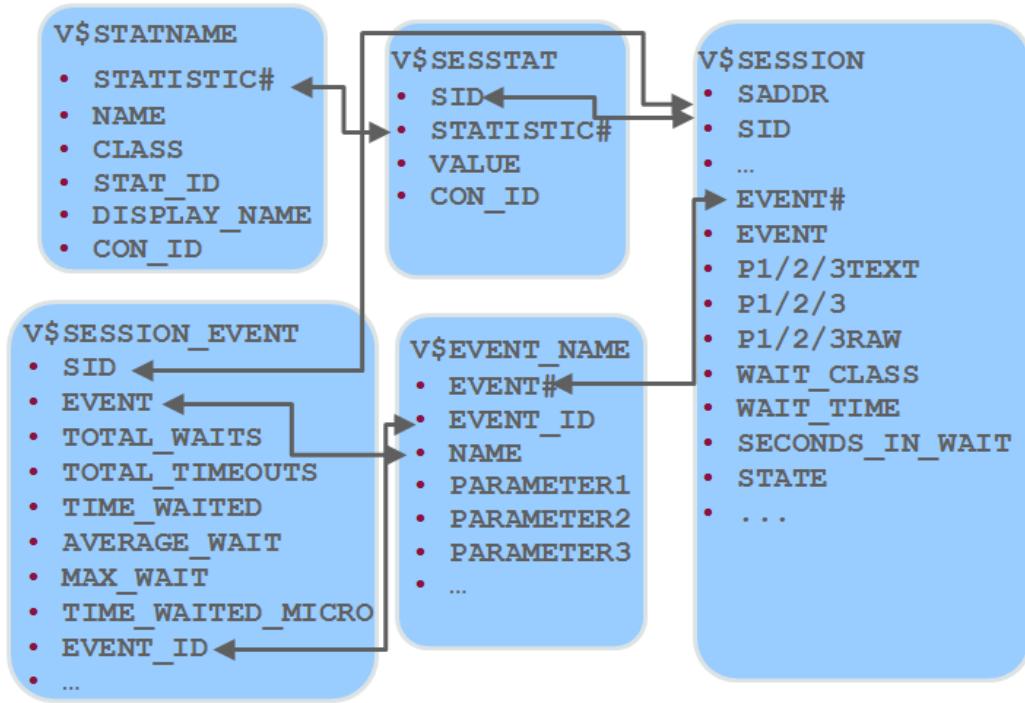
V\$LOCK
V\$UNDOSTAT
V\$WAITSTAT
V\$LATCH



You can respond to changes in performance only if you know the performance has changed. Oracle Database provides several ways to monitor the current performance of the database instance.

- **Enterprise Manager Database Express:** The database home page provides a quick check of the health of the instance and the server, with graphs showing CPU usage, active sessions, memory and data storage usage. The home page also shows any alerts that have been triggered. Additional detail is available on the Performance Hub page.
- **Performance views:** You can access these views directly with SQL*Plus. Occasionally, you may need to access these views for some detail about the raw statistics.

Monitoring Sessions



ORACLE®

You can display current session information for each user logged on by querying V\$SESSION. For example, you can use V\$SESSION to determine whether a session represents a user session, or was created by a database server process (background).

You can query either V\$SESSION or V\$SESSION_WAIT to determine the resources or events for which active sessions are waiting.

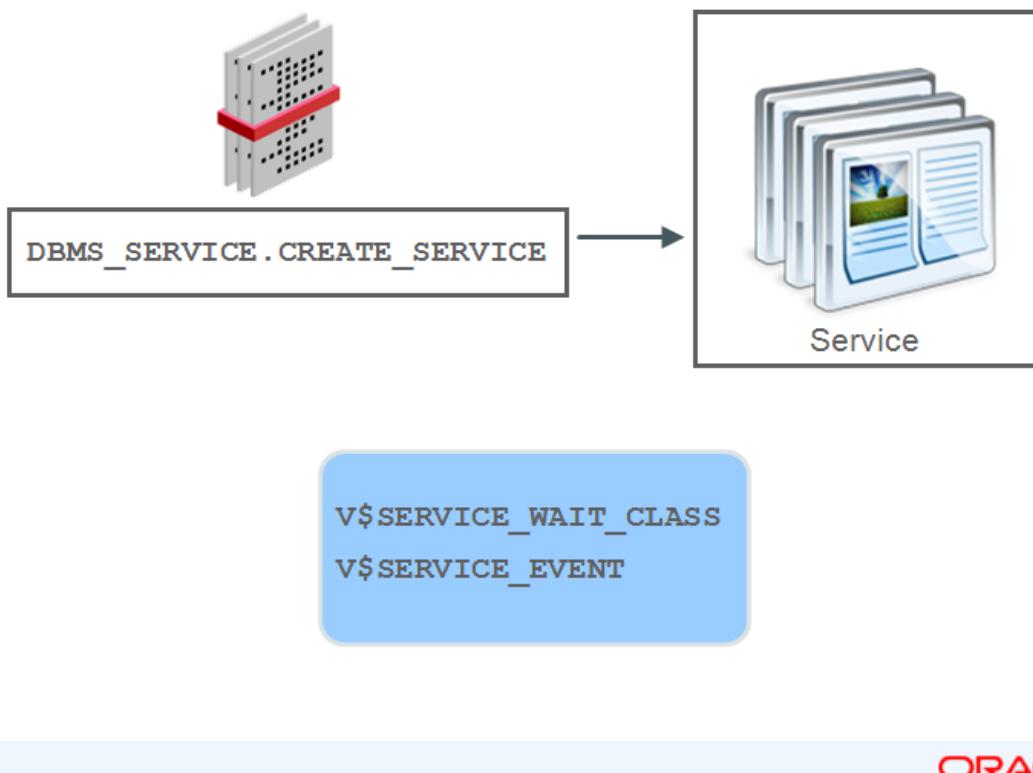
You can view user session statistics in V\$SESSTAT. The V\$SESSION_EVENT view lists information about waits for an event by a session.

Cumulative values for instance statistics are generally available through dynamic performance views, such as V\$SESSTAT and V\$SYSSTAT. Note that the cumulative values in dynamic views are reset when the database instance is shut down.

The V\$MYSTAT view displays the statistics of the current session.

You can also query V\$SESSMETRIC to display the performance metric values for all active sessions. This view lists performance metrics, such as CPU usage, number of physical reads, number of hard parses, and the logical read ratio.

Monitoring Services



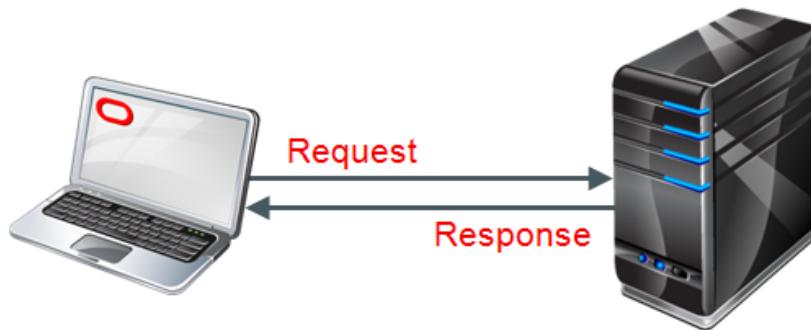
In a multi-tier environment where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately. Regardless of the session that was used for a particular request, if it connected via one of these services, the performance data of the session is captured under that service name.

You can define a service in the database by using the `DBMS_SERVICE` package and use the net service name to assign applications to a service.

Two views provide the same information that their like-named session counterparts provide, except that the information is presented at the service level rather than at the session level.

- `V$SERVICE_WAIT_CLASS` shows wait statistics for each service, broken down by wait class.
- `V$SERVICE_EVENT` shows the same information as `V$SERVICE_WAIT_CLASS`, except that it is further broken down by event ID.

Monitoring Wait Events



Wait events: Statistics indicating server process had to wait for an event to complete

V\$EVENT_NAME

ORACLE®

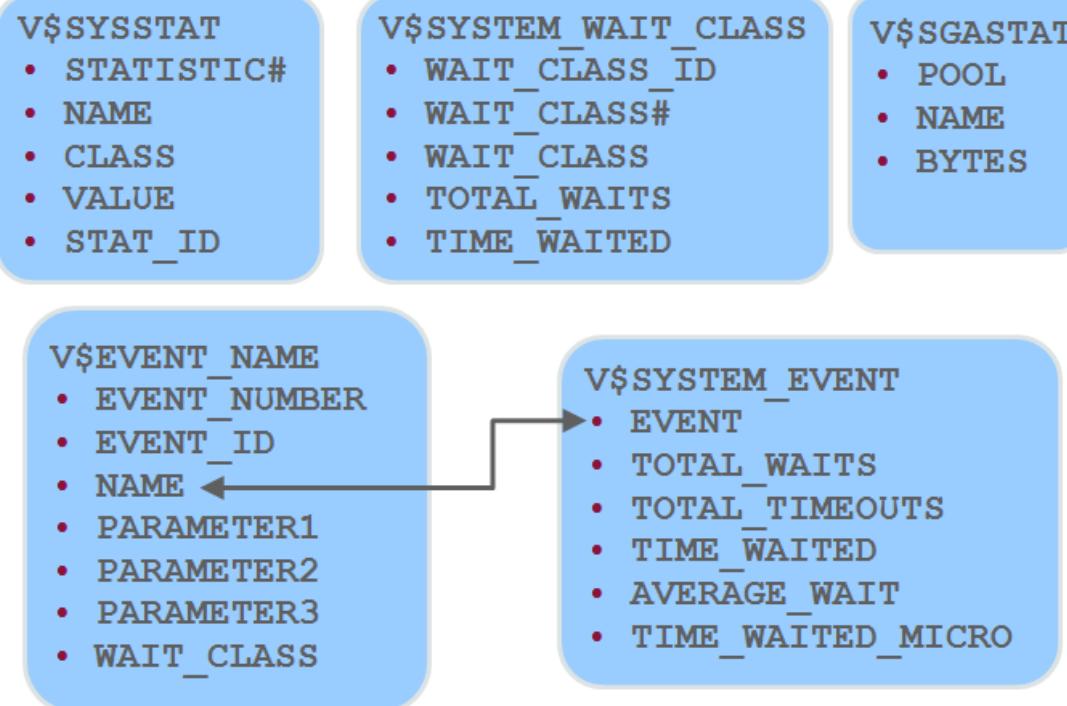
Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention. Remember that these are only symptoms of problems, not the actual causes.

A collection of wait events provides information about the sessions or processes that had to wait or must wait for different reasons.

Wait events are grouped into classes. The wait event classes include: Administrative, Application, Cluster, Commit, Concurrency, Configuration, Idle, Network, Other, Scheduler, System I/O, and User I/O.

Wait events are listed in the V\$EVENT_NAME view. There are more than 800 wait events in the Oracle Database, including free buffer wait, latch free, buffer busy waits, db file sequential read, and db file scattered read.

Viewing Instance Statistics



ORACLE®

To effectively diagnose performance problems, statistics must be available. The Oracle Database instance generates many types of cumulative statistics for the system, sessions, and individual SQL statements at the instance level. The Oracle Database server also tracks cumulative statistics on segments and services. When analyzing a performance problem in any of these scopes, you typically look at the change in statistics (delta value) over the period of time you are interested in.

Note: Instance statistics are dynamic and are reset at every instance startup. These statistics can be captured at a point in time and held in the database in the form of snapshots.

Wait Events Statistics

All the possible wait events are cataloged in the V\$EVENT_NAME view.

Cumulative statistics for all sessions are stored in V\$SYSTEM_EVENT, which shows the total waits for a particular event since instance startup.

When you are troubleshooting, you need to know whether a process has waited for any resource.

System-Wide Statistics

All the system-wide statistics are cataloged in the V\$STATNAME view: Over 400 statistics are available in Oracle Database.

The server displays all calculated system statistics in the V\$SYSSTAT view. You can query this view to find cumulative totals since the instance started.

System-wide statistics are classified by the tuning topic and the debugging purpose. The classes include general instance activity, redo log buffer activity, locking, database buffer cache activity, and so on. Each of the system statistics can belong to more than one class, so you cannot do a simple join on V\$SYSSTATS.CLASS and V\$SYSTEM_WAIT_CLASS.WAIT_CLASS#.

You can also view all wait events for a particular wait class by querying V\$SYSTEM_WAIT_CLASS.

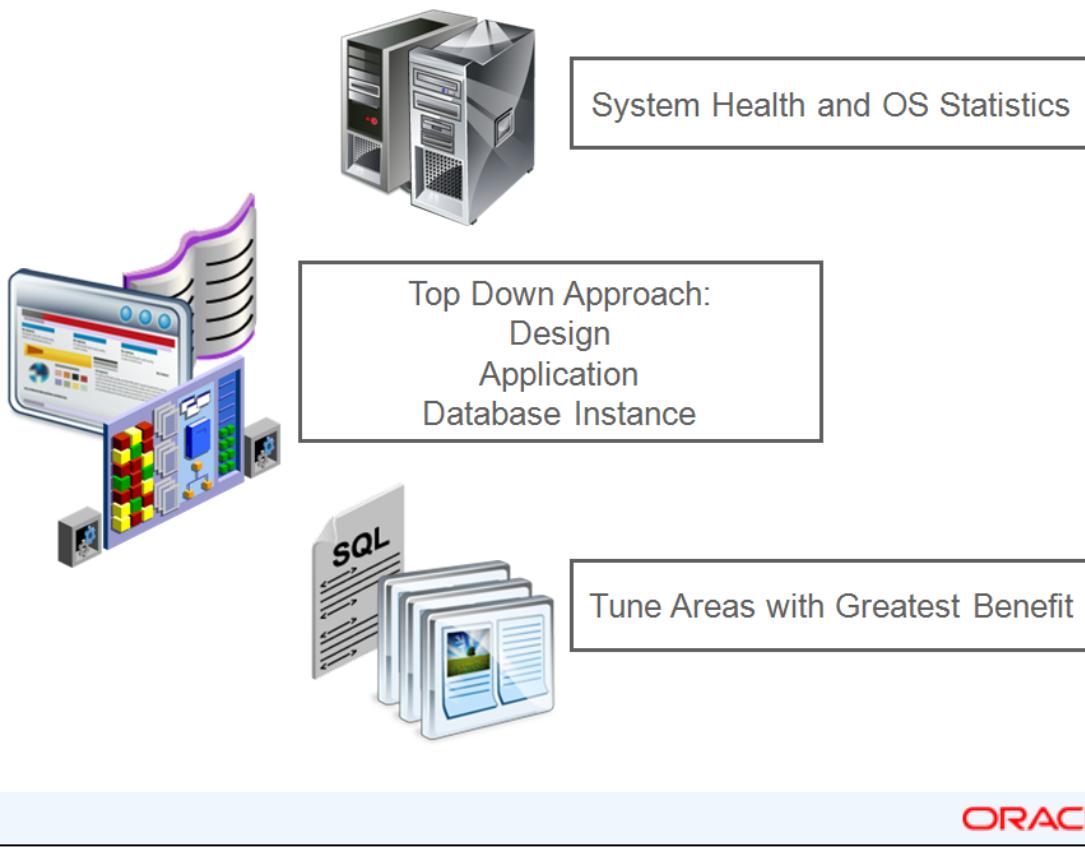
SGA Global Statistics

The server displays all calculated memory statistics in the V\$SGASTAT view. You can query this view to find cumulative totals of detailed SGA usage since the instance started.

When the STATISTICS_LEVEL parameter is set to BASIC, the value of the TIMED_STATISTICS parameter defaults to FALSE. Timing information is not collected for wait events and much of the performance-monitoring capability of the database is disabled. The explicit setting of TIMED_STATISTICS overrides the value derived from STATISTICS_LEVEL.

Performance Tuning Methodology

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



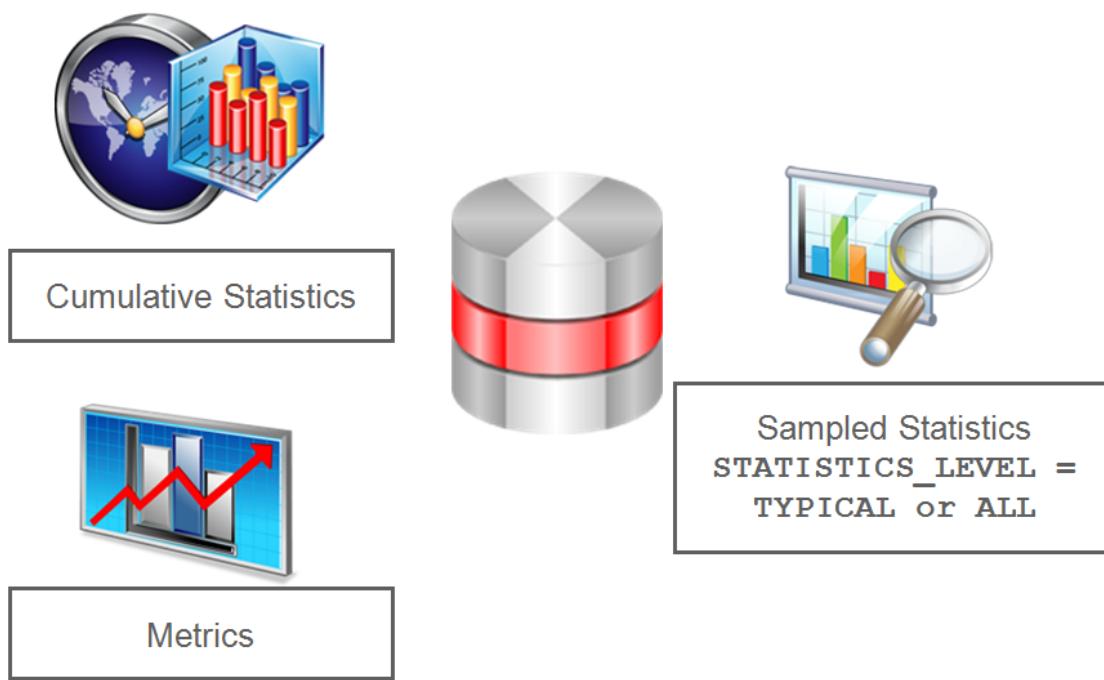
Oracle University and NxtTeam, Inc use only

Oracle has developed a tuning methodology based on years of experience. The basic steps are:

1. Check the OS statistics and general machine health before tuning the instance to be sure that the problem is in the database.
2. Tune from the top down. Start with the design, then the application, and then the instance. For example, try to eliminate full table scans that cause I/O contention before tuning the tablespace layout on disk. This activity often requires access to the application code.
3. Tune the area with the greatest potential benefit. The tuning methodology presented in this course is simple. Identify the biggest bottleneck and tune it. Repeat this step. All the various tuning tools have some way to identify the SQL statements, resource contention, or services that are taking the most time. The Oracle database provides a time model and metrics to automate the process of identifying bottlenecks. The Advisors available in Oracle Database use this methodology.
4. Stop tuning when you meet your goal. This step implies that you set tuning goals.

This is a general approach to tuning the database instance and may require multiple passes.

Database Server Statistics and Metrics



ORACLE®

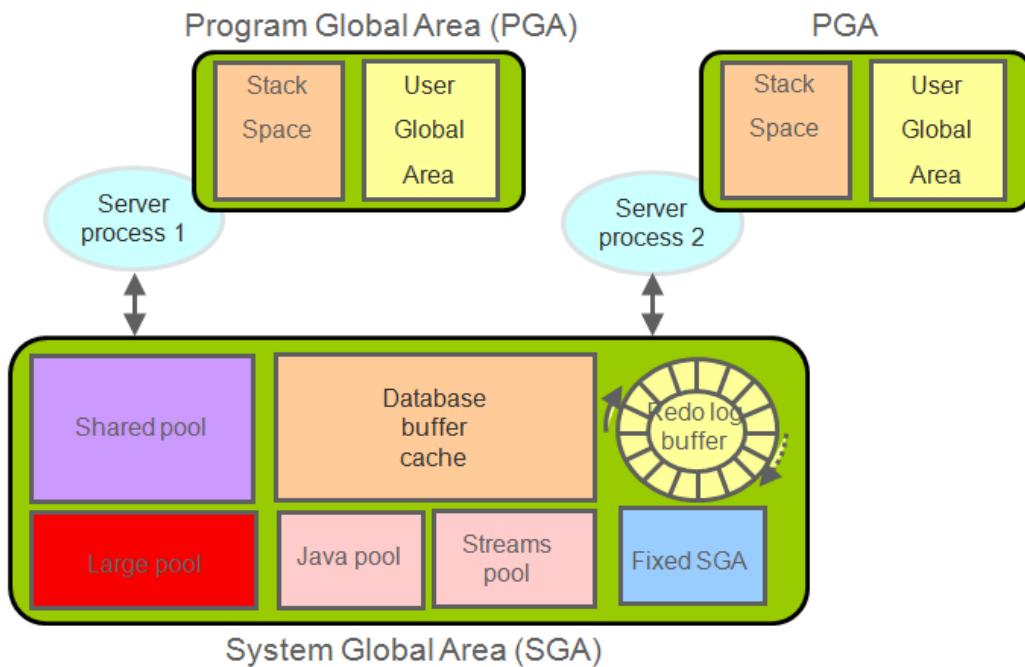
The Oracle Database server software captures information about its own operation. Three major types of data are collected: cumulative statistics, metrics, and sampled statistics.

Cumulative statistics are counts and timing information of a variety of events that occur in the database server. Some are quite important, such as buffer busy waits. Others have little impact on tuning, such as index block split. The most important events for tuning are usually the ones showing the greatest cumulative time values. The statistics in Oracle Database are correlated by the use of a time model. The time model statistics are based on a percentage of DB time, giving them a common basis for comparison.

Metrics are statistic counts per unit. The unit could be time (such as seconds), transaction, or session. Metrics provide a base to proactively monitor performance. You can set thresholds on a metric, causing an alert to be generated. For example, you can set thresholds for when the reads per millisecond exceed a previously recorded peak value or when the archive log area is 95% full.

Sampled statistics are gathered automatically when `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`. Sampled statistics allow you to look back in time. You can view session and system statistics that were gathered in the past, in various dimensions, even if you had not thought of specifying data collection for these beforehand.

Managing Memory Components



ORACLE®

Because there is a finite amount of memory available on a database server and an Oracle Database instance, you must pay attention to how memory is allocated. If too much memory is allowed to be used by a particular area that does not need it, other areas may not function properly because of lack of memory. With the ability to have memory allocation automatically determined and maintained for you, the task is simplified greatly. But even automatically tuned memory needs to be monitored for optimization and may need to be manually configured to some extent.

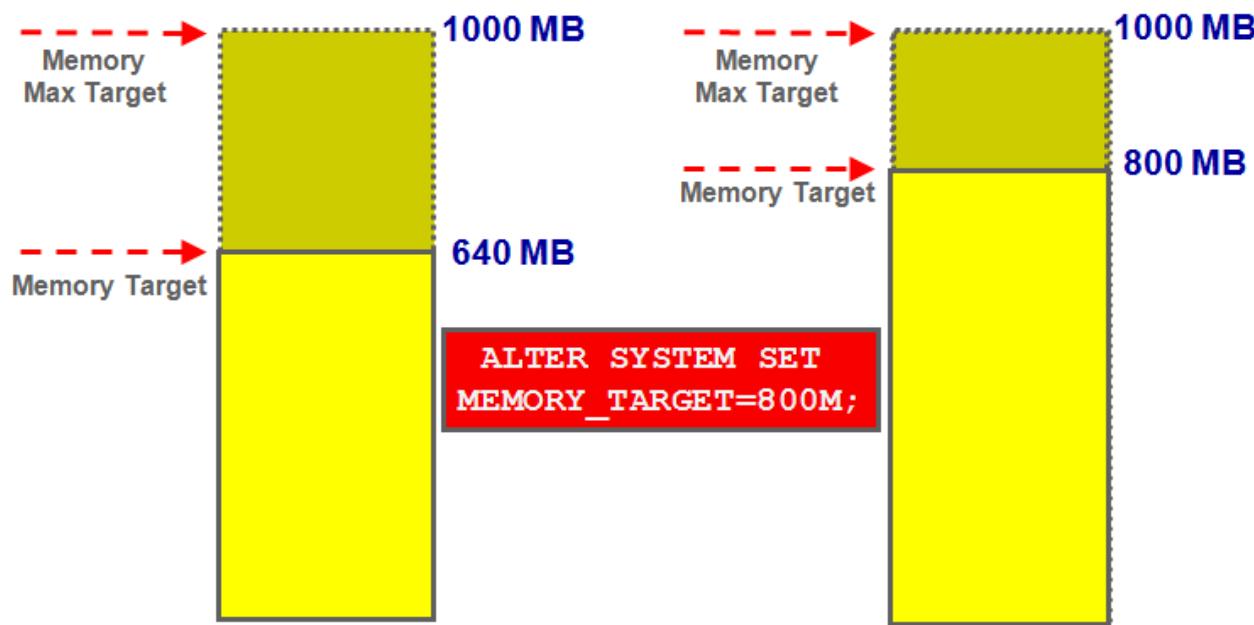
Guidelines for efficient memory usage include the following.

- **Fit the SGA into physical memory:** If possible, it is best to fit the SGA into physical memory, which provides the fastest access. Even though the OS may provide additional virtual memory, this memory, by its nature, can often be swapped out to disk. On some platforms, you can use the `LOCK_SGA` initialization parameter to lock the SGA into physical memory. This parameter cannot be used in conjunction with Automatic Memory Management (AMM) or Automatic Shared Memory Management (ASMM).
- **Tune for the most efficient use of memory:** When a SQL statement executes, data blocks are requested for reading or writing, or both. This is considered a logical I/O. As the block is requested, the block is checked to see whether it already exists in memory. If it is not in memory, it is read from disk, which is called a physical I/O. When the block is found in memory, the cost is several orders of magnitude less than the cost of reading the block from disk. The size of the SGA components in combination with the workload has a large affect on the number of physical reads. A simple view of this implies that you should increase the memory for the SGA components as much as possible. A larger SGA is not always better. There is a point where adding more memory yields diminishing returns. This principle applies to the buffer cache, the shared pool, and other SGA components. In practice, you find that shifting memory from one SGA component to another may increase overall performance, without changing the total amount of memory given to the SGA depending on the characteristics of the workload. Memory has an upper limit in all current machines. That limit may be imposed

by the hardware, operating system, or the cost of the memory. The goal of memory tuning is to produce the most efficient use of existing memory. When the workload changes often, the most efficient division of memory between the SGA components will change. There is also the amount of memory allocated to SGA and PGA. Online transaction processing (OLTP) systems typically use very little PGA memory compared to data warehouse (DW) or decision support systems (DSS). Using the existing memory efficiently also includes tuning the applications. A poorly tuned application can use large quantities of memory. For example, an application that uses frequent full table scans because indexes do not exist or are unusable can cause a large amount of I/O, reducing performance. The first and most effective tuning technique is to tune high cost SQL statements.

Enterprise Manager Cloud Control and Enterprise Manager Database Express both provide Memory Advisors. These tools monitor the memory usage by the SGA components and PGA, and project the differences in terms of efficiency for increased and decreased memory allocations. These projections use the current workload. They can help you size the SGA based on the activity in your particular database. The advisors make sizing recommendations for manual settings, when the automatic memory management is disabled. These same advisors provide input to automatic memory management, to determine the most efficient component sizes.

Automatic Memory Management



ORACLE®

Automatic Memory Management (AMM) allows the Oracle Database server to manage SGA memory and instance PGA memory sizing automatically. To do so (on most platforms), you set only a target memory size initialization parameter (`MEMORY_TARGET`) and a maximum memory size initialization parameter (`MEMORY_MAX_TARGET`), and the database server dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands.

With this memory management method, the database server also dynamically tunes the sizes of the individual SGA components and the sizes of the individual PGAs.

Because the target memory initialization parameter is dynamic, you can change the target memory size at any time without restarting the database instance. The maximum memory size serves as an upper limit so that you cannot accidentally set the target memory size too high. Because certain SGA components either cannot easily shrink or must remain at a minimum size, the database server also prevents you from setting the target memory size too low.

This indirect memory transfer relies on the operating system (OS) mechanism of freeing shared memory. After memory is released to the OS, the other components can allocate memory by requesting memory from the OS. Currently, Automatic Memory Management is implemented on Linux, Solaris, HPUX, AIX, and Windows.

Oracle recommends the use of AMM unless you have special requirements.

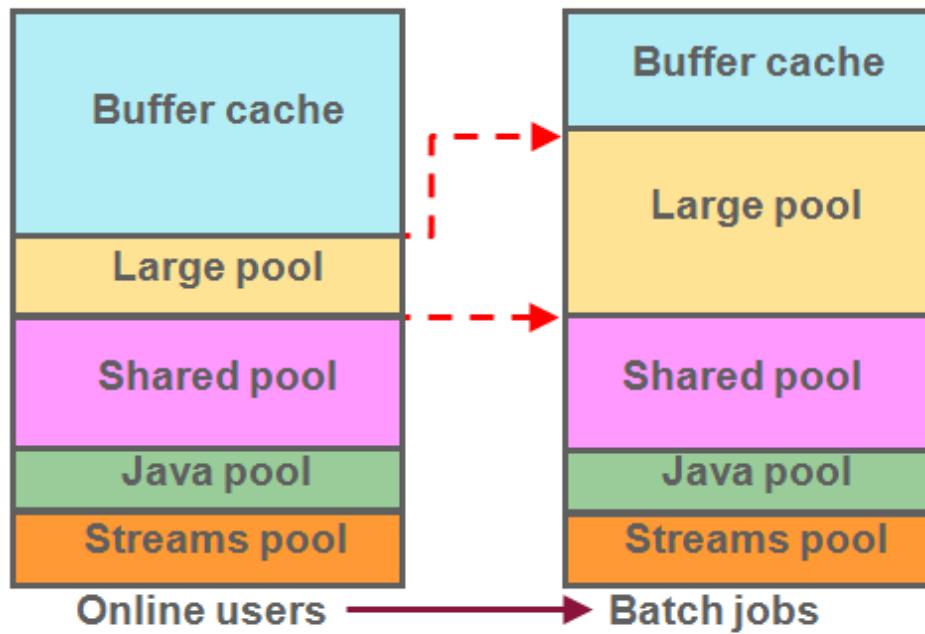
Use the following views to monitor Automatic Memory Management:

- **V\$MEMORY_DYNAMIC_COMPONENTS:** Shows the current sizes of all dynamically tuned memory components, including the total sizes of the SGA and instance PGA

- **V\$MEMORY_RESIZE_OPS:** Shows a circular history buffer of the last 800 memory resize requests
- **V\$MEMORY_TARGET_ADVICE:** Provides tuning advice for the MEMORY_TARGET initialization parameter

Automatic Shared Memory Management

Example:



ORACLE®

If you need a fixed PGA, consider the use of *Automatic Shared Memory Management (ASMM)*, which also simplifies SGA memory management. You specify the total amount of SGA memory available to an instance by using the `SGA_TARGET` initialization parameter and Oracle Database server automatically distributes this memory among the various SGA components to ensure the most effective memory utilization.

For example, in a system that runs large online transactional processing (OLTP) jobs during the day (requiring a large buffer cache) and runs parallel batch jobs at night (requiring a large value for the large pool), you would have to simultaneously configure both the buffer cache and the large pool to accommodate your peak requirements.

With ASMM, when the OLTP job runs, the buffer cache uses most of the memory to allow for good I/O performance. When the data analysis and reporting batch job starts up later, the memory is automatically migrated to the large pool so that it can be used by parallel query operations without producing memory overflow errors.

The Oracle Database server remembers the sizes of the automatically tuned components across instance shutdowns if you are using a server parameter file (SPFILE). As a result, the system needs to learn the characteristics of the workload again each time an instance is started. It can begin with information from the past instance and continue evaluating the workload where it left off at the last shutdown.

The Automatic Shared Memory Management feature uses the SGA memory broker that is implemented by two background processes: Manageability Monitor (MMON) and Memory Manager (MMAN). Statistics and memory advisory data are periodically captured in memory by MMON. MMAN coordinates the sizing of the memory components according to MMON decisions. The SGA memory broker keeps track of the sizes of the components and pending resize operations.

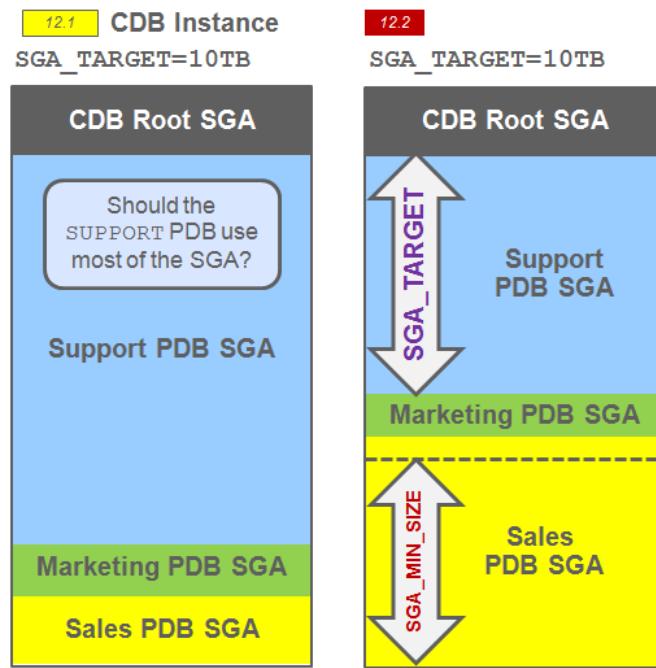
The SGA memory broker observes the system and workload in order to determine the ideal distribution of memory. It performs this check every few minutes so that memory can always be present where needed. In the absence of Automatic Shared Memory Management, components had to be sized to anticipate their individual worst-case memory requirements.

On the basis of workload information, Automatic Shared Memory Management:

- Captures statistics periodically in the background
- Uses memory advisors
- Performs what-if analysis to determine the best distribution of memory
- Moves memory to where it is most needed
- Saves component sizes after shutdown if an SPFILE is used (the sizes can be resurrected from before the last shutdown)

Managing the SGA for PDBs

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASS ONLY. COPYING OR MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



ORACLE®

In a CDB, there is one SGA allocation for the instance, shared by all containers including the CDB root and all PDBs. Most of the SGA is a cache that favors frequently-accessed objects in the buffer cache, the shared pool and the in-memory column store. In Oracle Database 12c Release 1, active PDBs dominate the space in the SGA cache. In the graphic, the Support PDB has an active memory-intensive workload. It monopolizes the SGA. The Marketing PDB needs very little SGA. Sales PDB performance depends on critical buffer cache data and parsed cursors. The Support PDB is more active and is evicting its data.

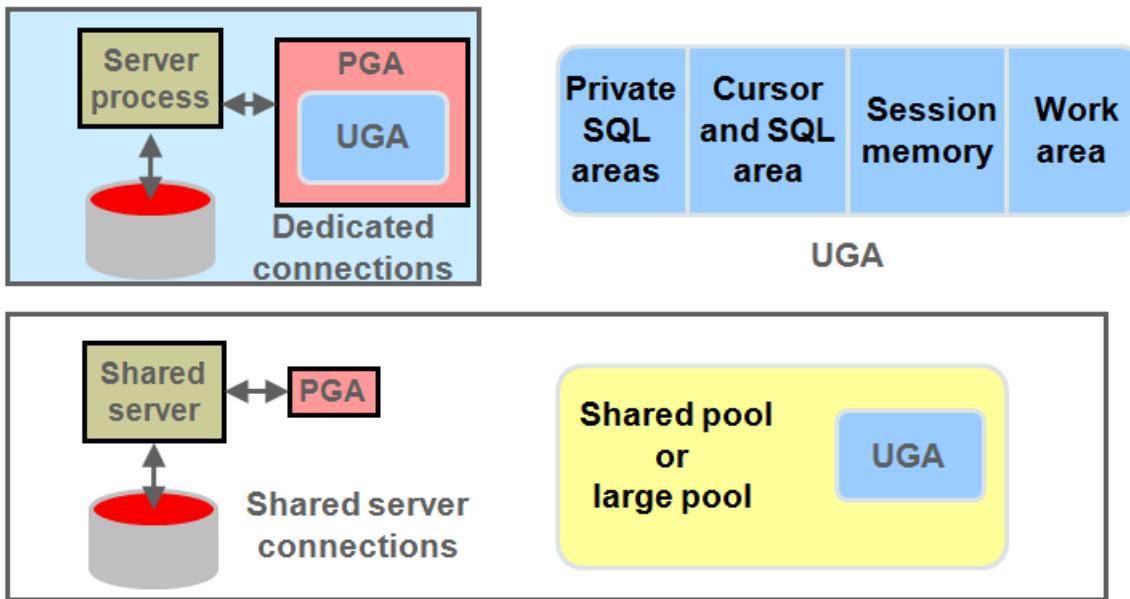
Oracle Database 12c Release 2 provides SGA and PGA memory management at the PDB level:

- Setting an `SGA_TARGET` for a PDB enforces a hard limit for the PDB's SGA and provides more SGA for the other containers within the CDB. The sum of all PDB `SGA_TARGET` parameter values does not necessarily need to be less than the instance `SGA_TARGET` value, but each PDB `SGA_TARGET` value cannot exceed the instance `SGA_TARGET` nor `SGA_MAX_SIZE`. `SGA_TARGET` for PDBs works only if the CDB's `SGA_TARGET` parameter is set.
- Setting `DB_CACHE_SIZE` and `SHARED_POOL_SIZE` guarantees minimum sizes for the PDB.
- Setting an `SGA_MIN_SIZE` for a PDB guarantees the SGA space for the PDB.

`SGA_TARGET`, `DB_CACHE_SIZE`, and `SHARED_POOL_SIZE` do not work if `MEMORY_TARGET` is set for the CDB.

No more than 50% of the memory can be set aside for the PDB minimums: `SGA_MIN_SIZE`, `DB_CACHE_SIZE`, and `SHARED_POOL_SIZE`.

Managing the Program Global Area



ORACLE®

The *program global area (PGA)* is a memory region that contains data and control information for a server process. It is nonshared memory created by the Oracle server when a server process is started. Access to it is exclusive to that server process. The total PGA memory allocated by all server processes attached to an Oracle instance is also referred to as the aggregated PGA memory allocated by the instance.

Part of the PGA can be located in the SGA when using shared servers.

PGA memory typically contains the following:

- **Private SQL Area:** A private SQL area, also called the user global area (UGA) contains data, such as bind information and runtime memory structures. This information is specific to each session's invocation of the SQL statement; bind variables hold different values, and the state of the cursor is different, among other things. Each session that issues a SQL statement has a private SQL area. Each user that submits the same SQL statement has his or her own private SQL area that uses a single shared SQL area. Thus, many private SQL areas can be associated with the same shared SQL area. The location of a private SQL area depends on the type of connection established for a session. If a session is connected through a dedicated server, private SQL areas are located in the server process's PGA. However, if a session is connected through a shared server, part of the private SQL area is kept in the SGA.
- **Cursor and SQL Areas:** The application developer of an Oracle Pro*C program or Oracle Call Interface (OCI) program can explicitly open cursors or handles to specific private SQL areas, and use them as a named resource throughout the execution of the program. Recursive cursors that the database issues implicitly for some SQL statements also use shared SQL areas.
- **Work Area:** For complex queries (for example, decision support queries), a big portion of the PGA is dedicated to work areas allocated by memory-intensive operators. A sort operator uses a work area (the sort area) to

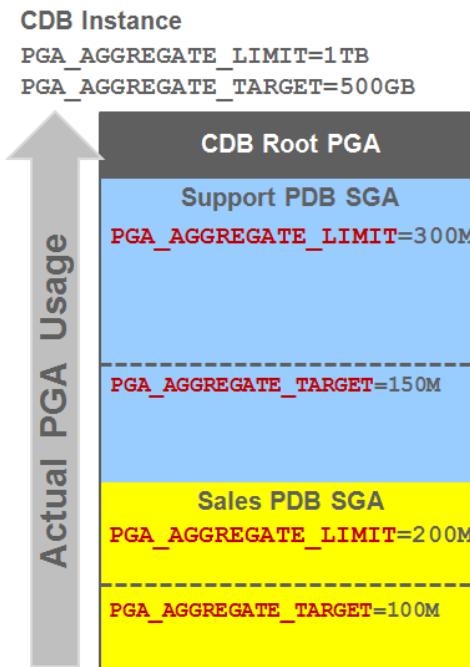
perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (the hash area) to build a hash table from its left input. The size of a work area can be controlled and tuned. Generally, bigger work areas can significantly improve the performance of a particular operator at the cost of higher memory consumption.

- **Session Memory:** Session memory is the memory allocated to hold a session's variables (logon information) and other information related to the session. For a shared server, the session memory is shared and not private.

By default, Oracle Database automatically and globally manages the total amount of memory dedicated to the instance PGA. You can control this amount by setting the initialization parameter `PGA_AGGREGATE_TARGET`. Oracle Database then tries to ensure that the total amount of PGA memory allocated across all database server processes and background processes never exceeds this target. But this is a target value and not a hard limit.

`PGA_AGGREGATE_LIMIT` sets a hard limit for the amount of PGA that can be used. The minimum value is 1024 MB and the maximum is 120% of physical memory minus the total SGA, and it must be at least as large as `PGA_AGGREGATE_TARGET`. If `PGA_AGGREGATE_LIMIT` is not set, it defaults to 200% of `PGA_AGGREGATE_TARGET` within the same minimum and maximum as stated. When `PGA_AGGREGATE_LIMIT` is exceeded, the sessions using the most memory will have their calls aborted. Parallel queries will be treated as a unit. If the total PGA memory usage is still over the limit, sessions using the most memory will be terminated. SYS processes and fatal background processes are exempt from this limit.

Managing the PGA for a PDB



ORACLE®

In Oracle Database 12c Release 2 you can set the following the parameters at the PDB level:

- **PGA_AGGREGATE_TARGET:** Specifies the target aggregate PGA memory available to all server processes attached to the instance
- **PGA_AGGREGATE_LIMIT:** Specifies a hard limit for aggregate PGA memory

Summary for Lesson 11

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

In this lesson, you should have learned to:

- Describe the activities that you perform to manage database performance
- Use performance views and tools to monitor database instance performance
- Describe the Oracle performance tuning methodology
- Describe statistics and metrics that are collected by the Oracle Database server
- Configure and monitor memory components for optimal performance



ORACLE®

Practice 11 Overview

- 11-1: Managing Performance
- 11-2: Resolving Lock Issues

ORACLE®

Practice 11-1 Managing Performance in EM Express

Overview

In this practice, you manage the performance of the database instance by using Enterprise Manager Database Express (EM Express).

You could use V\$ views to analyze performance statistics and metrics, but it is much easier to use EM Express or EM Cloud Control. Whichever tool you use, the key to identifying instance performance issues are wait events and high cost SQL.

Note: In [Practice 14-2 Using ADDM](#), you use EM Cloud Control to monitor and diagnose instance performance.

Assumptions

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

Tasks

Start an Application Workload

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. Execute the `$HOME/labs/PERF_setup_tuning.sh` shell script. This script recreates PDB1 and PDB2. In PDB1, it creates a user called `admin_pdb1`, a tablespace called `TBS_APP`, and a schema called `oe` in the `TBS_APP` tablespace. It does the same thing in PDB2, except it creates a user called `admin_pdb2`. This script takes a minute to run. Ignore the error messages.

```
$ $HOME/labs/PERF_setup_tuning.sh

...
$
```

3. Start an application workload in PDB1 and PDB2. **Note:** This script generates continuous output in the terminal window where it starts.

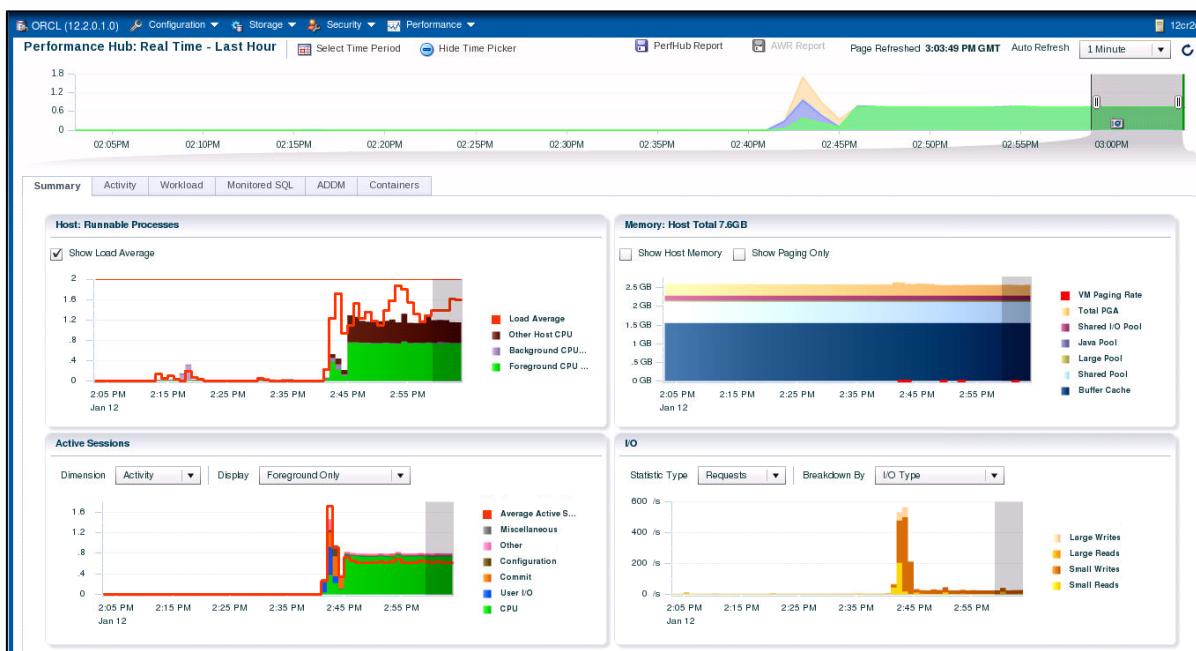
```
$ $HOME/labs/PERF_loop.sh

...
```

Review the Performance Hub in EM Express

1. Connect the client browser to the CDB in EM Express.

- a. Open a Firefox web browser, and enter the URL <https://localhost:5500/em> .
- b. If a "This Connection is Untrusted" dialog is displayed, click **I Understand the Risks**, click **Add Exception**, and then click **Confirm Security Exception** to proceed.
- c. On the Login page, enter the user name **sys** and the password as specified in [Appendix D - Product-Specific Credentials](#). Leave the Container Name box empty, select the **as sysdba** check box, and click **Login**.
2. Select **Performance**, and then **Performance Hub**. The Performance Hub provides a consolidated view of all performance data for a given time range. You must have the Oracle Diagnostics Pack (paid option) to use the Performance Hub.



3. Learn about the Time Picker at the top of the page. The Time Picker displays average active sessions over time.
 - a. Click **Hide Time Picker** and then **Show Time Picker** to hide and show it.
 - b. At the top of the page, click **Select Time Period**. In the dialog box, you can select a time range, and the detail tabs will display the available performance data for the selected time range. Click the drop-down list and review the options. Notice that you can choose to view historical and real-time data.
 - c. Select **Real Time - Last Hour**, and click **OK**. In real-time mode, performance data is retrieved from in-memory views. The time picker shows data for the past hour and you can select any time range from within this period. The default selection is the past 5 minutes.
 - d. If there are peaks in the time picker, on the chart you can drag the selected time range to the period of interest to get more information. Drag the time picker to test this out, and try moving just one of the time picker handles to increase and decrease the time range.
 - e. Click **Select Time Period**, select **Historical - Day**, and click **OK**. In historical mode, data is retrieved from the Automatic Workload Repository (AWR). You can select any time period, provided there is sufficient data in AWR. When you switch to historical mode, the default selected time range is dependent

- on the amount of data shown in the time picker: if the time picker displays data for the past week, the default selected time range is one day; and if the time picker displays data for the past day, the default selected time range is one hour.
- f. Change the time picker back to displaying the last hour.
 - g. Click **PerfHub Report** at the top of the page. Notice that you save the contents of the tabs with details for top SQL statements. Click **Cancel**.
4. The Performance Hub organizes performance data by dividing it into different tabs. Each tab addresses a specific aspect of database performance.
- a. The **Summary** tab, which is currently displayed, is available in both real-time and historical mode. In real-time mode, this tab shows metrics data that gives an overview of system performance in terms of host resource consumption (CPU, I/O and memory), and average active sessions. In historical mode, this tab displays system performance in terms of resource consumption, average active sessions, and load profile information.
 - b. Click the **Activity** tab. This tab displays Active Session History (ASH) analytics, and is available in both real-time and historical mode.
 - c. Click the **Workload** tab. This tab is available in both real-time and historical mode, and shows metric information about the workload profile, such as call rates, logon rate, and the number of sessions. It also displays the Top SQL for the selected time range. In real-time mode, this tab displays top SQL only by database time, but in historical mode, you can also display top SQL by other metrics, such as CPU time or executions.
 - d. Click the **Monitored SQL** tab. This tab displays monitored executions of SQL, PL/SQL, and database operations, and is available in both real-time and historical mode.
 - e. Click the **ADDM** tab. This tab displays the performance findings and recommendations of Automatic Database Diagnostic Monitor (ADDM) for database tasks performed in the selected time period. It is available in both real-time and historical mode. The ADDM analyzes data in the AWR to identify potential performance bottlenecks.
 - f. Click the **Containers** tab. This tab displays performance information about each PDB in the CDB, including active sessions, memory used, I/O requests, and I/O throughput.

[View Wait Statistics on the Activity Tab](#)

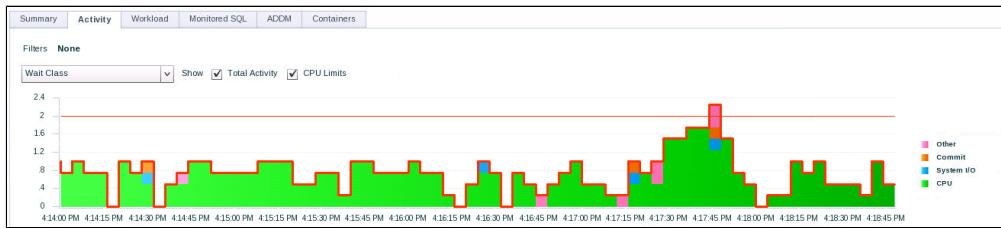
On the Activity tab, you can view the Active Session History (ASH). ASH is part of the Diagnostics and Tuning Pack. It samples information from the [G]V\$ views allowing you to see current and historical information about active sessions on the database. An active session is a session that is waiting on CPU or any event that does not belong to the IDLE wait class.

1. If your workload script has ended, start it again in the terminal window.

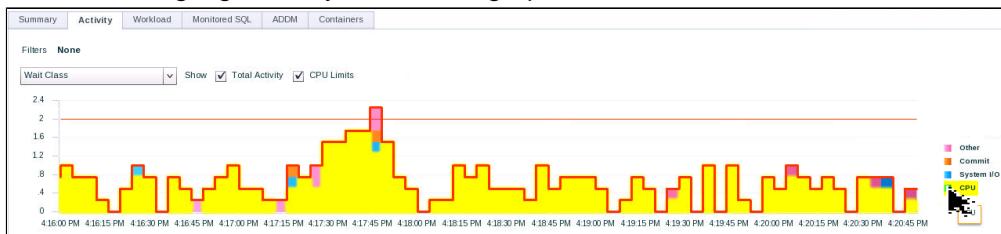
```
$ $HOME/labs/PERF_loop.sh  
...
```

2. In EM Express, click the **Refresh** button a few times. Eventually you will get some data in the time picker.
3. Click the **Activity** tab.
4. View the graph in the middle of the page, which shows wait event information. Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Waits and the associated SQL are key indicators for determining the root cause of an issue.
 - a. By default, the graph displays the average active session waits by time, filtered by class (notice that Wait Class is selected in the filters drop-down list). Each wait class consists of wait events. For example, waits

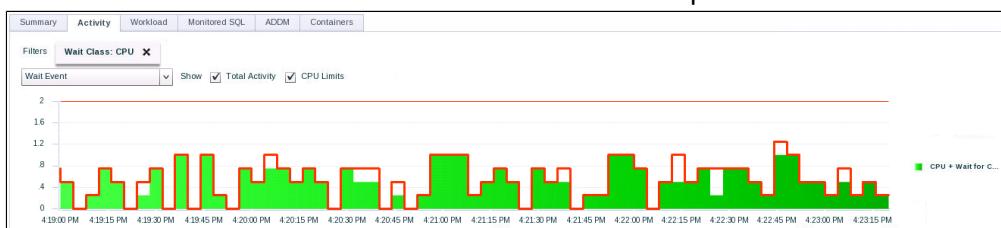
resulting from DBA commands that cause users to wait (for example, an index rebuild) are in the Administrative class. Another example is the System I/O class which consists of waits for background process IO; for example, a DBWR wait for 'db file parallel write.' The classes are listed in the legend to the right of the graph.



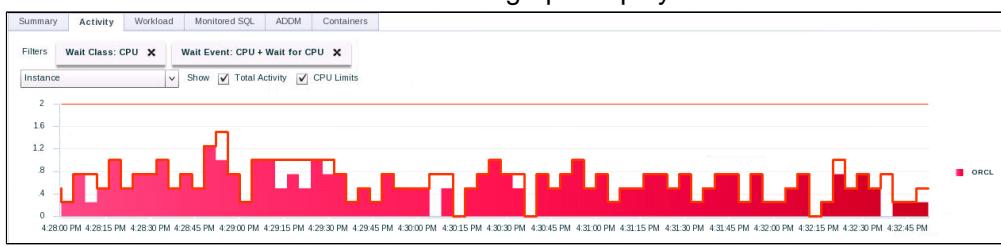
- b. Position your cursor over one of the wait classes in the legend, for example, the **CPU** class. Notice that the class is highlighted in yellow in the graph.



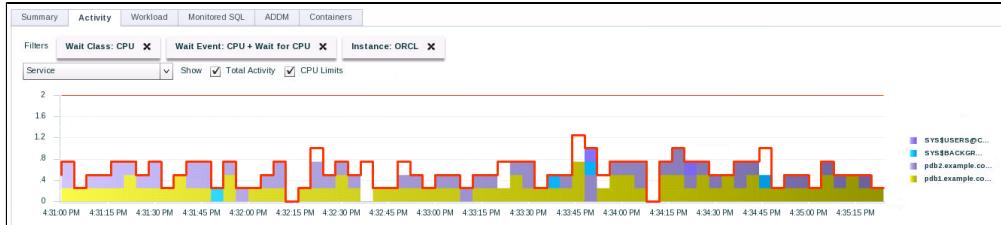
- c. In the legend, click the **CPU** wait class. A filter is created and the graph drills down into the different wait events for the CPU wait class. Notice that the filters drop-down list is now Wait Event.



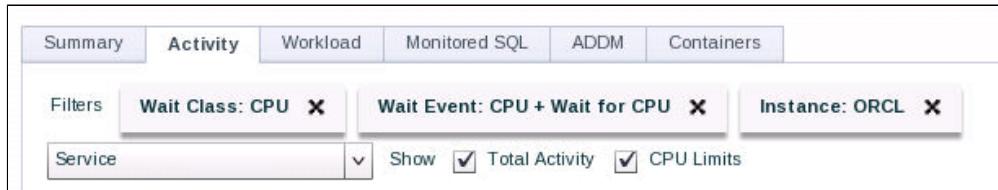
- d. Click the graph. The graph is displaying only one item at the moment, so don't worry about clicking the wrong part. Notice when you positioned your cursor over the graph it turned yellow again. Clicking the graph further drills down into wait events. So you can either click the legend items or click the graph items themselves to drill down. Now the graph displays the waits for the ORCL instance.



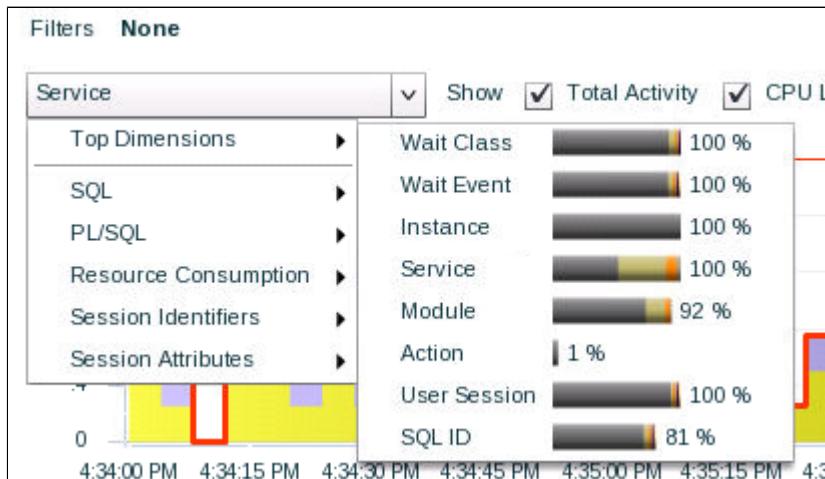
- e. In the legend, click **ORCL**. You have now drilled down into the waits per service.



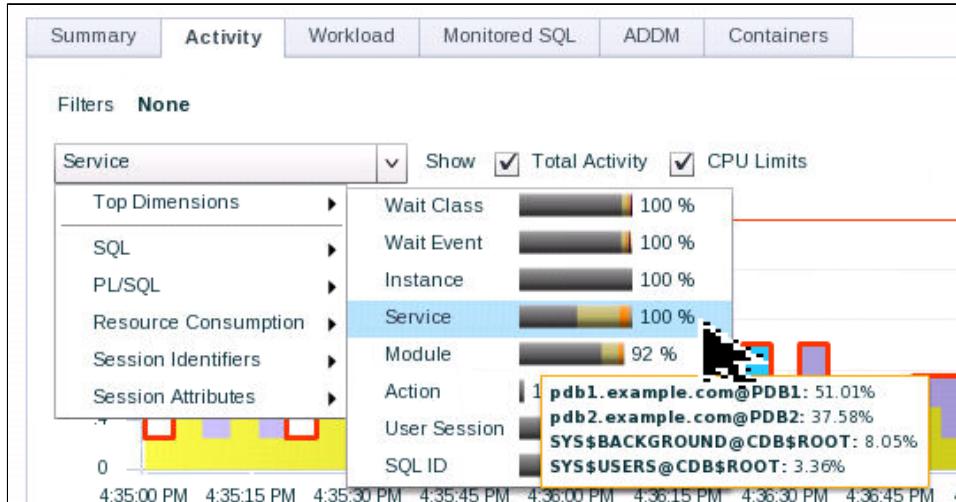
- f. Remove the filters in the graph by clicking the Xs for each filter.



- g. Another way to filter the graph is to select a filter in the drop-down list. In the drop-down list, select **Top Dimensions**. Notice that the names of the top dimensions are the same names you just saw as you drilled down into the graph through the legend; for example, Wait Class, Wait Event, Instance, Service, and so on. Selecting a top dimension is a quick and easy way to jump directly to a particular drill-down level.



- h. Position your cursor over **Service** and view the percentage breakdown for service waits.

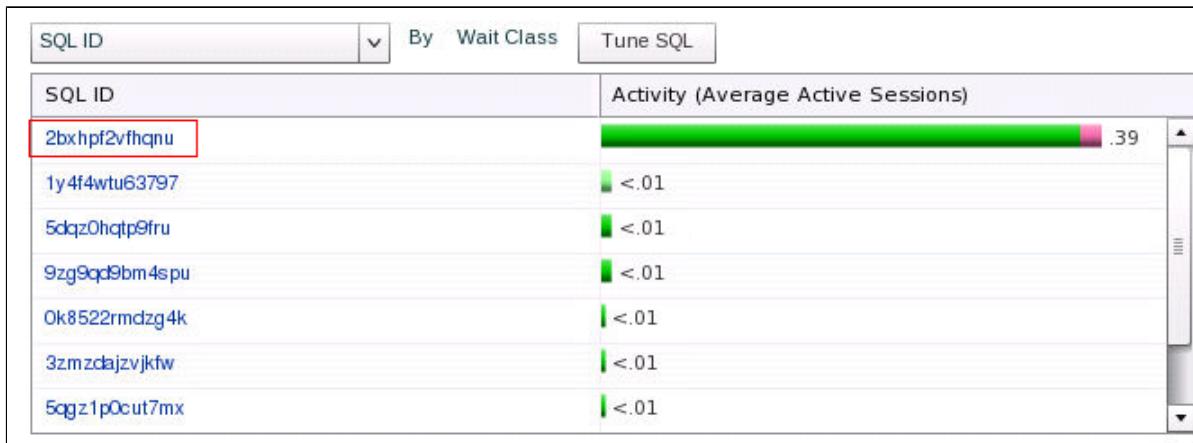


Filter Wait Statistics for a SQL ID

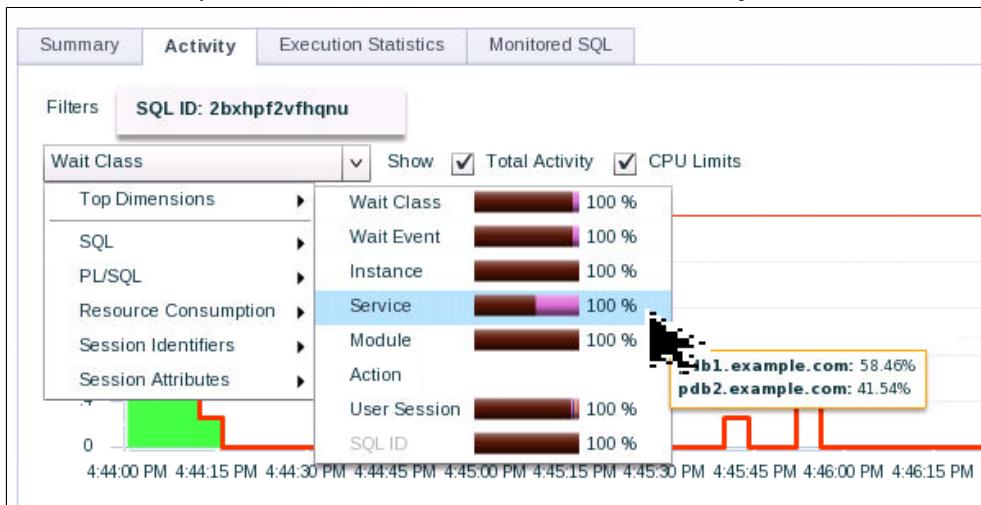
1. If your workload script has ended, start it again in the terminal window.

```
$ $HOME/labs/PERF_loop.sh
...
$
```

2. In EM Express, at the bottom left, view the table that shows the activity (average active sessions) for each SQL ID. Click the SQL ID (in this example, the SQL ID is 2bxhpf2vhqnu) that has the greatest average (the top one). The Summary tab is displayed with performance information about the selected SQL ID.

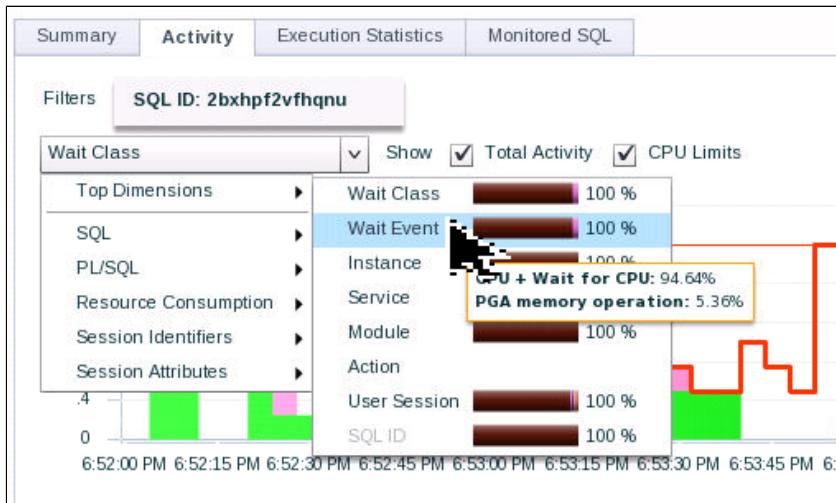


3. Click the **Activity** tab. Notice that the Activity tab is filtered based on your selected SQL ID.
 4. In the filters drop-down list, select **Wait Class**, and then **Top Dimensions**. Position your cursor over **Service**.



5. Question: What does the information tell you?
 Answer: In this example, the percentage value is higher for PDB1 (58.46%) than PDB2 (41.54%), which means that executions in PDB1 have to wait longer than in PDB2. Your values will be different.

6. Position your cursor over **Wait Event**. Notice that the execution of the statement is waiting for CPU resources.



7. Click **Log Out** to log out of EM Express, and close the browser window.
8. In the terminal window, press **Ctrl+C** to stop the `PERF_loop.sh` script, and close the terminal window.

Practice 11-2 Resolving Lock Issues

Overview

In this practice, two users try to update data for an employee at the same time in SQL*Plus and cause a lock conflict. Using Enterprise Manager Database Express (EM Express), you detect the blocking session, and then resolve the conflict by killing the blocking session.

Note: You can also use EM Cloud Control to monitor locks conflicts.

Assumptions

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

Tasks

Create a Lock Conflict

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window 1.

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

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

2. Execute the `$HOME/labs/RESOLVE_LOCK_ISSUES_setup.sh` shell script. You can ignore the error messages.

```
$ $HOME/labs/RESOLVE_LOCK_ISSUES_setup.sh

...
$
```

3. Start SQL*Plus and connect to `PDB1` as `NGREENBERG`. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus ngreenberg/<password>@PDB1

...
SQL>
```

4. Update the `HR.EMPLOYEES` table, but do not commit or exit the SQL*Plus session. The following statement updates the phone number to `650.555.1212` for employee ID 110. This session will be referred to as the "blocking session."

```
SQL> UPDATE hr.employees SET phone_number='650.555.1212' WHERE employee_id =
110;

1 row updated.

SQL>
```

5. Open another terminal window, and source the oraenv script. For the ORACLE_SID value, enter ORCL . This window will be referred to as window 2.

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

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

6. In window 2, start SQL*Plus and connect as SMAVRIS. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus smavris/<password>@PDB1

...
SQL>
```

7. Update the HR.EMPLOYEES table. The following statement updates the salary to 8300 for employee ID 110.

```
SQL> UPDATE hr.employees SET salary = 8300 WHERE employee_id = 110;
```

8. Notice that the session in window 2 is hung. This session will be referred to as the "blocked session." Leave this session as is and move on to the next step.

9. Question: Which situation can cause lock conflicts?

Answer: The most common cause of lock conflicts is an uncommitted change, but there are a few other possible causes, such as:

- **Long-running transactions:** Many applications use batch processing to perform bulk updates. These batch jobs are usually scheduled for times of low or no user activity, but in some cases, they may not have finished or may take too long to run during the low activity period. Lock conflicts are common when transaction and batch processing are being performed simultaneously.
- **Unnecessarily high locking levels:** Not all databases support row-level locking (Oracle added support for row-level locks in 1988 with release 6). Some databases still lock at the page or table level. Developers writing applications that are intended to run on many different databases often write their applications with artificially high locking levels so that Oracle Database behaves similarly to these less capable database systems. Developers who are new to Oracle also sometimes unnecessarily code in higher locking levels than are required by Oracle Database.

10. SMAVRIS, who is connected in window 2, informs you that his transaction is blocked.

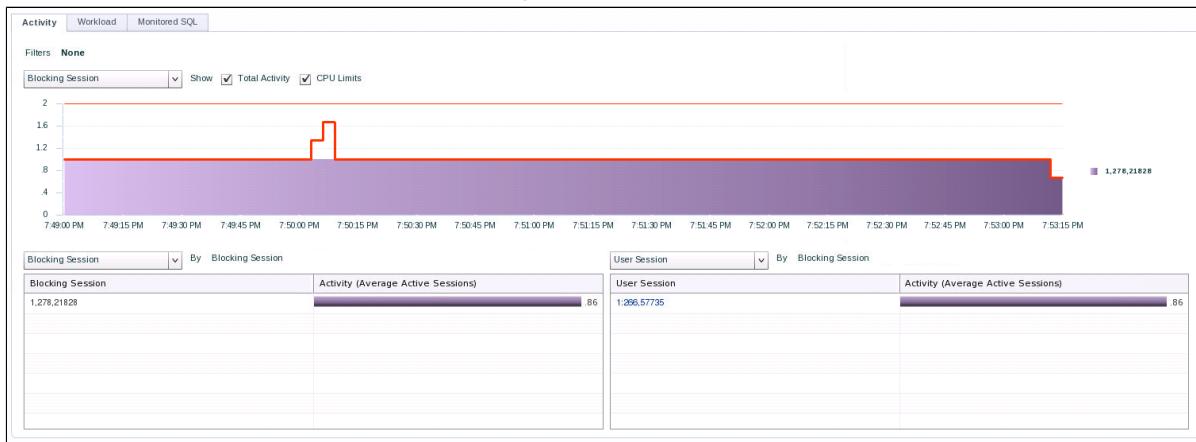
Question: What is the best way to fix the locking conflict?

Answer: The solution is to have the session release the lock. Contact the user and ask that the transaction be completed. In an emergency, it is possible for the administrator to terminate the session holding the lock. Remember that when a session is killed, all work within the current transaction is lost (rolled back). A user whose session is killed must log in again and redo all work since the killed session's last commit.

Release the Lock in EM Express

In this section, you use EM Express to locate the blocking session and retrieve the identifiers that enable you to kill the blocking session.

1. Connect the client browser to PDB1 in EM Express.
 - a. Open a Firefox web browser, and enter the URL <https://localhost:5500/em>.
 - b. On the Login page for EM Express, enter the user name **sys** and the password as specified in [Appendix D - Product-Specific Credentials](#). Enter **PDB1** as the container name. Select the **as sysdba** check box. Click **Login**.
2. Select **Performance**, and then **Performance Hub**. The Activity tab is displayed by default.
3. In the filter drop-down list, select **Resource Consumption**, and then **Blocking Session**.
4. In the bottom drop-down list, also select **Resource Consumption**, and then **Blocking Session** to retrieve the identifiers that allow you to kill the blocking session.

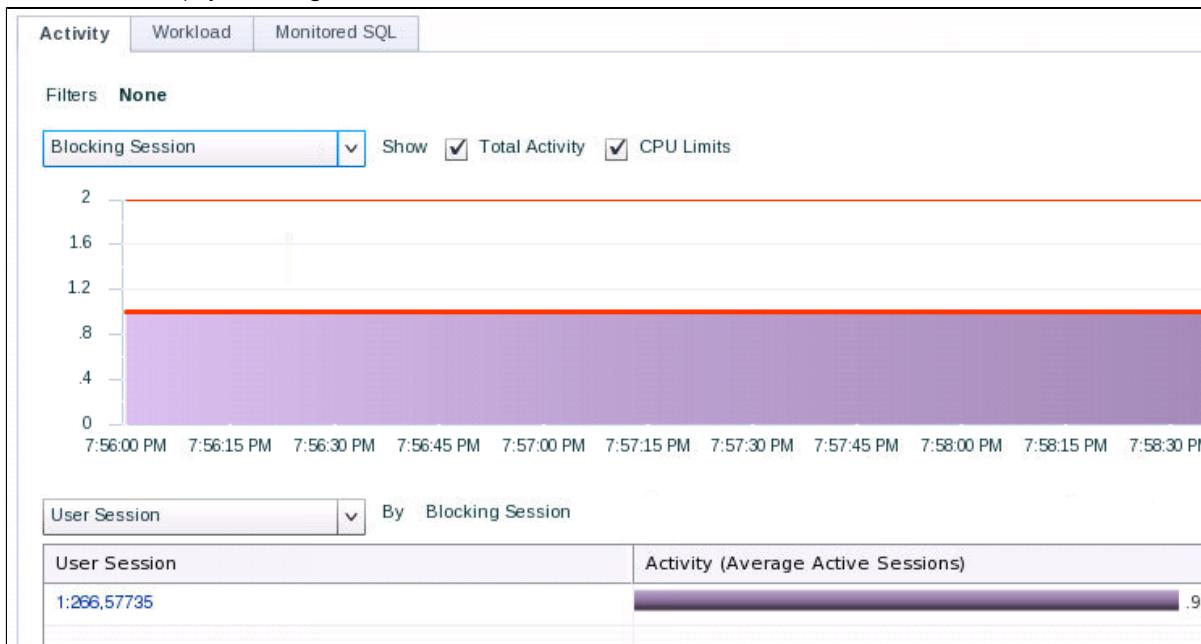


5. In the Blocking Session column at the bottom of the page, make note of the second number (SID) and third number (serial number). The blocking session is identifiable by its SID and serial number, and you will use this information later in SQL*Plus. In this example, the SID is 278 and the serial number is 21828. Your numbers will be different.

The screenshot shows the same table from the previous step, enclosed in a rectangular box. The table has a single row with the following data:

Blocking Session
1,278,21828

6. Be aware that if, in the bottom drop-down list, you select User Session (select **Top Dimensions**, and then **User Session**), you will get the list of blocked sessions and therefore not the correct session to kill.



7. Log out of EM Express and close the browser window.

Kill the Blocking Session

With the SID and serial number in hand, you can now kill the blocking session in SQL*Plus.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window 3.

```
$ . 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 `SYSTEM` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB1

...
SQL>
```

3. Kill the session using the SID and serial number that you identified in EM Express. Your numbers will be different than those shown below.

```
SQL> ALTER SYSTEM KILL SESSION '278,21828';
```

```
System altered.  
SQL>
```

4. In window 2, find out what happened to the blocked session. That is, verify whether the salary was updated to 8300.

- a. Notice that in window 2, the blocked session is now released and the result is "1 row updated."

```
1 row updated.  
SQL>
```

- b. Query the HR.EMPLOYEES table to find out whether the salary was updated for employee ID 110. The result shows that the salary was updated to 8300.

```
SQL> SELECT phone_number , salary FROM hr.employees WHERE employee_id =  
110;
```

PHONE_NUMBER	SALARY
515.124.4269	8300

```
SQL>
```

- c. Rollback the salary update because you don't need to keep the update.

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

5. In window 1, find out what happened to the blocking session.

- a. Notice that in window 1, the result "1 row updated" is also displayed.

```
1 row updated.  
SQL>
```

- b. Query the HR.EMPLOYEES table to find out the phone number for employee ID 110. The result indicates that the session was killed. Therefore, the locks were released for other sessions and the update transaction for the phone number was rolled back.

```
SQL> SELECT phone_number FROM hr.employees WHERE employee_id = 110;  
SELECT phone_number FROM hr.employees WHERE employee_id = 110  
*  
ERROR at line 1:  
ORA-00028: your session has been killed  
SQL>
```

6. Exit SQL*Plus in all terminal windows and close the windows.

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

12

Tuning SQL



ORACLE®

Objectives for Lesson 12

After completing this lesson, you should be able to:

- Describe the SQL tuning methodology
- Manage optimizer statistics
- Use the SQL Tuning Advisor to identify and tune SQL statements that are using the most resources
- Use the SQL Access Advisor to tune a workload



ORACLE®

SQL Tuning Process

1. Identify poorly tuned SQL statements.

2. Tune the individual SQL statements.

3. Tune the application as a whole.



ORACLE®

SQL tuning process:

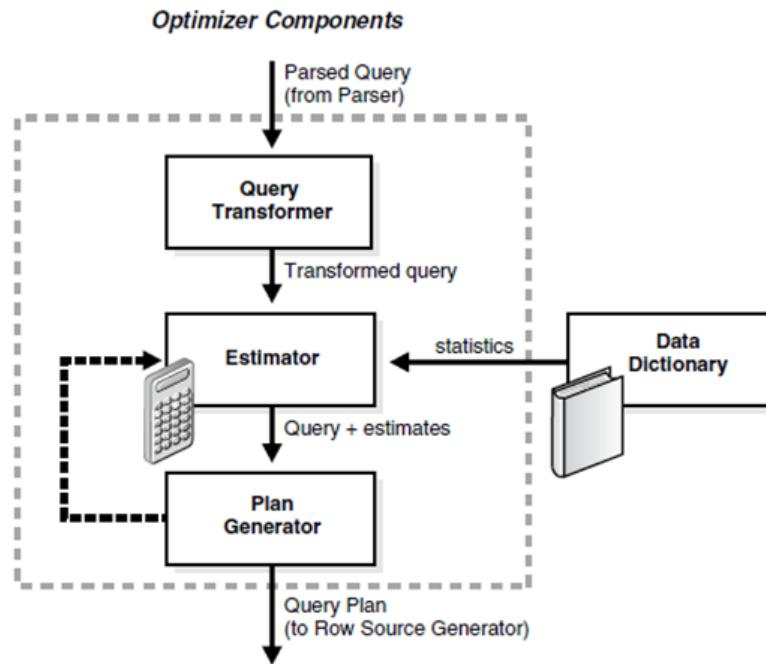
1. **Identify poorly tuned SQL statements.** Generally, the tuning effort that yields the most benefit is SQL tuning. Poorly tuned SQL uses more resources than required. This inefficiency prevents scalability, uses more OS and database resources, and increases response time. To tune poorly tuned SQL statements, they must be identified, and then tuned. SQL statements can be tuned individually, but often the solution that optimizes one statement can hurt the performance of several others. The SQL statements that use the most resources are by definition the statements in need of tuning. These are statements that have the longest elapsed time, use the most CPU, or do the most physical or logical reads.
2. **Tune the individual statements.** Tune the individual statements by checking the optimizer statistics, check the explain plan for the most efficient access path, test alternate SQL constructions, and test possible new indexes, materialized views, and partitioning.
3. **Tune the application as a whole.** Test the application as a whole, using the tuned SQL statements. Is the overall performance better?

The methodology is sound, but tedious. Tuning an individual statement is not difficult. Testing the overall impact of the individual statement tuning on an application can be very difficult.

In Oracle Database, a set of SQL advisors are available to identify and tune statements, individually, or as a set.

Oracle Optimizer

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



ORACLE

The *optimizer* is the part of the Oracle Database server that creates the execution plan for a SQL statement. The determination of the execution plan is an important step in the processing of any SQL statement and can greatly affect execution time.

The execution plan is a series of operations that are performed in sequence to execute the statement. The optimizer considers many factors related to the referenced objects and the conditions specified in the query. The information necessary to the optimizer includes:

- Statistics gathered for the system (I/O, CPU, and so on) as well as schema objects (number of rows, index, and so on)
- Information in the dictionary
- WHERE clause qualifiers
- Hints supplied by the developer

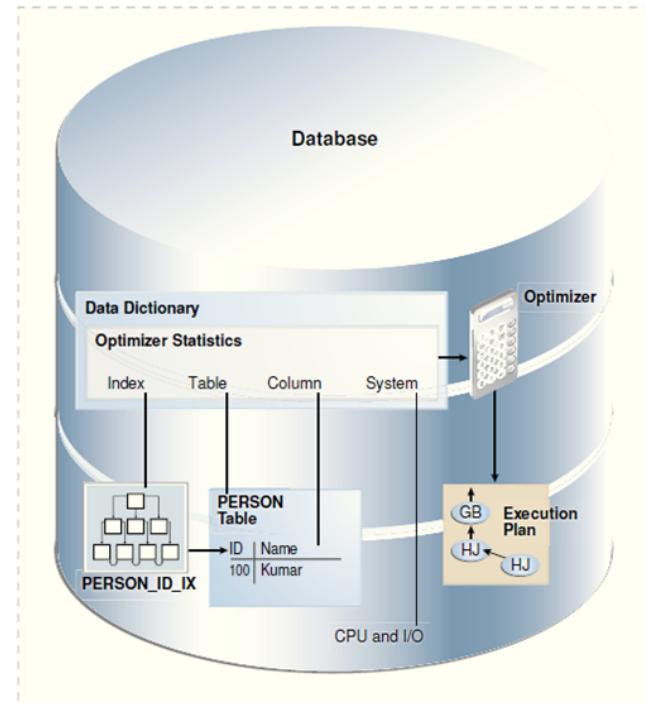
The optimizer:

- Evaluates expressions and conditions
- Uses object and system statistics
- Decides how to access the data and join tables
- Determines the most efficient path

When you use diagnostic tools, such as Enterprise Manager, EXPLAIN PLAN, and SQL*Plus AUTOTRACE, you can see the execution plan that the optimizer chooses.

Optimizer Statistics

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING OR DISTRIBUTING THESE MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



ORACLE®

Optimizer statistics include table, column, index, and system statistics. Statistics for tables and indexes are stored in the data dictionary. These statistics are not intended to provide real-time data. They provide the optimizer a statistically correct snapshot of data storage and distribution, which the optimizer uses to make decisions on how to access data.

The statistics that are collected include:

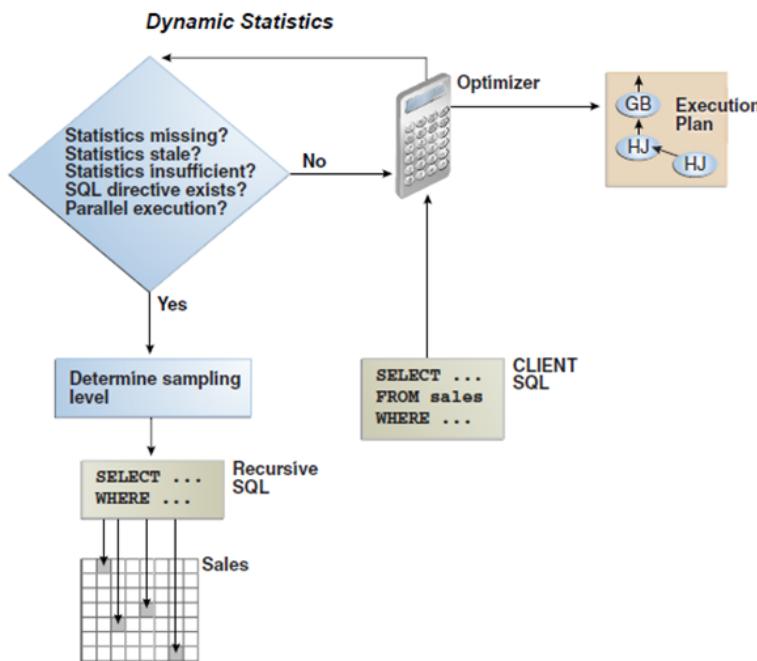
- Size of the table or index in database blocks
- Number of rows
- Average row size and chain count (tables only)
- Height and number of deleted leaf rows (indexes only)

As data is inserted, deleted, and modified, these statistics change. Because the performance impact of maintaining real-time data distribution statistics is prohibitive, these statistics are updated by periodically gathering statistics on tables and indexes.

Optimizer statistics are collected automatically by an automatic maintenance job that runs during predefined maintenance windows once daily by default. System statistics are operating system characteristics that are used by the optimizer. These statistics are not collected automatically. For details about collecting system statistics, see the Oracle Database Performance Tuning Guide.

Optimizer statistics are not the same as the database performance statistics that are gathered in the AWR snapshot.

Optimizer Statistics Collection



ORACLE®

Optimizer statistics are collections of data that are specific details about database objects. These statistics are essential for the query optimizer to choose the best execution plan for each SQL statement. These statistics are gathered periodically and do not change between gatherings.

Statistics can be collected in the following ways:

- Automatically: Automatic Maintenance Tasks
- Manually: DBMS_STATS package
- By setting database initialization parameters
- By importing statistics from another database

The recommended approach to gathering optimizer statistics is to allow the Oracle Database server to automatically gather the statistics. The Automatic Maintenance Tasks can be created automatically at database creation time and is managed by the Scheduler. It gathers statistics on all objects in the database that have either missing or stale optimizer statistics by default. You can change the default configuration through the Automatic Maintenance Tasks page.

System statistics describe the system's hardware characteristics, such as I/O and CPU performance and utilization, to the query optimizer. When choosing an execution plan, the optimizer estimates the I/O and CPU resources required for each query. System statistics enable the query optimizer to more accurately estimate I/O and CPU costs, and thereby choose a better execution plan. System statistics are collected using the

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS CLASSROOM ONLY IS PROHIBITED.

DBMS_STATS.GATHER_SYSTEM_STATS procedure. When the Oracle Database server gathers system statistics, it analyzes system activity in a specified period of time. System statistics are not automatically gathered. Oracle Corporation recommends that you use the DBMS_STATS package to gather system statistics.

If you choose not to use automatic statistics gathering, you must manually collect statistics in all schemas, including system schemas. If the data in your database changes regularly, you also need to gather statistics regularly to ensure that the statistics accurately represent characteristics of your database objects. To manually collect statistics, use the DBMS_STATS package. This PL/SQL package is also used to modify, view, export, import, and delete statistics.

You can also manage optimizer and system statistics collection through database initialization parameters, for example:

- The OPTIMIZER_DYNAMIC_SAMPLING parameter controls the level of dynamic sampling performed by the optimizer. You can use dynamic sampling to estimate statistics for tables and relevant indexes when they are not available or are too out of date to trust. Dynamic sampling also estimates single-table predicate selectivity when collected statistics cannot be used or are likely to lead to significant errors in estimation.
- The STATISTICS_LEVEL parameter controls all major statistics collections or advisories in the database and sets the statistics collection level for the database. The values for this parameter are BASIC, TYPICAL, and ALL. You can query the V\$STATISTICS_LEVEL view to determine which parameters are affected by the STATISTICS_LEVEL parameter.

Note: Setting STATISTICS_LEVEL to BASIC disables many automatic features and is not recommended.

Setting Optimizer Statistics Preferences

DBMS_STATS.GATHER_*_STATS procedures: Gather statistics for an entire database or for individual objects using default values

Use the SET_*_PREFS procedures to create preference values for any object that is not owned by SYS or SYSTEM

Query DBA_TAB_STAT_PREFS to view object-level preferences

Execute the DBMS_STATS.GET_PREFS procedure for each preference to see the global preferences



The DBMS_STATS.GATHER_*_STATS procedures can be called at various levels to gather statistics for an entire database or for individual objects, such as tables. When the GATHER_*_STATS procedures are called, several of the parameters are often allowed to default. The supplied defaults work well for most of the objects in the database, but for some objects or schemas, the defaults need to be changed. Instead of running manual jobs for each of these objects, Oracle Database enables you to set values (called preferences) for individual objects, schemas, or databases, or to change the default values with a global-level command.

The preferences specify the parameters that are given to the gather procedures. The SET_*_PREFS procedures create preference values for any object that is not owned by SYS or SYSTEM. The expected use is that the DBA will set the global preferences for any parameters that should be database-wide. These will be applied for any parameter that is allowed to default.

The SET_DATABASE_PREFS procedure iterates over all the tables and schemas in the database setting the specified preference. SET_SCHEMA_PREFS iterates over the tables in the specified schema. SET_TABLE_PREFS sets the preference value for a single table.

All object preferences—whether set at the database, schema, or table level—are held in a single table. Changing the preferences at the schema level overwrites the preferences that were previously set at the table level.

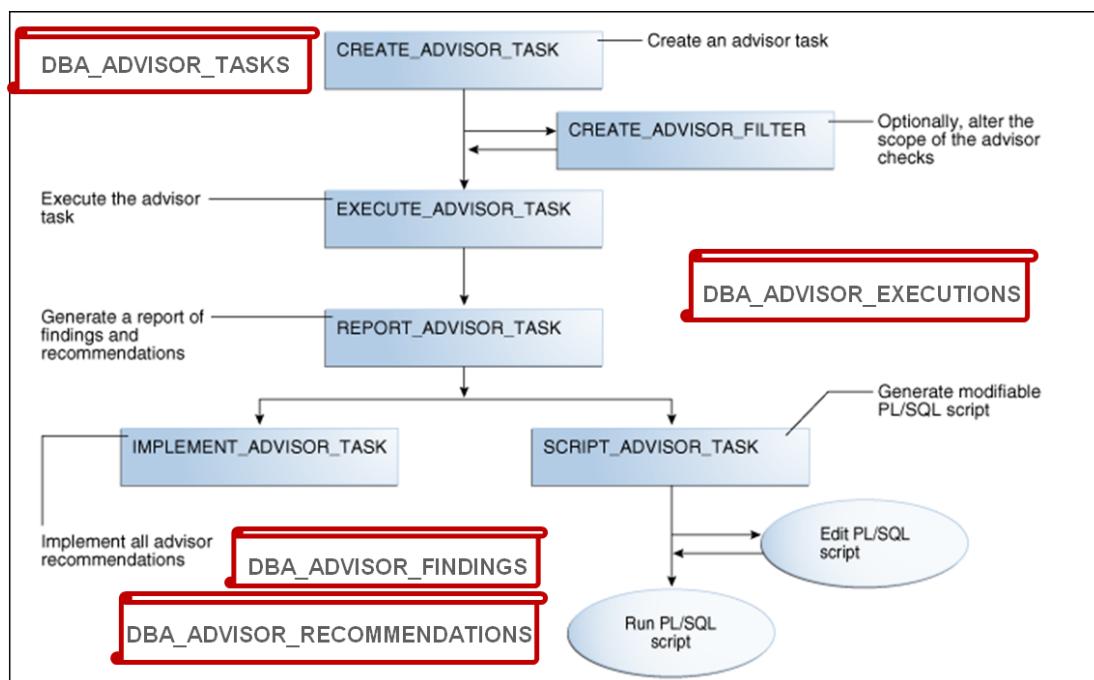
When the various gather procedures execute, they retrieve the object-level preferences that were set for each object. You can view the object-level preferences in the DBA_TAB_STAT_PREFS view. Any preferences that are not set at the object level will be set to the global-level preferences. You can see the global preferences by calling the DBMS_STATS.GET_PREFS procedure for each preference.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

For details about these preferences, see the DBMS_STATS documentation in the Oracle Database PL/SQL Packages and Types Reference.

Preferences may be deleted with the DBMS_STATS.DELETE_*_PREFS procedures at the table, schema, and database levels. You can reset the global preferences to the recommended values with the DBMS_STATS.RESET_PARAM_DEFAULTS procedure.

Optimizer Statistics Advisor



ORACLE®

Since the introduction of the cost-based optimizer, optimizer statistics play a significant part in determining the execution plan for queries. Therefore, it is critical for the optimizer to have accurate and up-to-date statistics. The DBMS_STATS package serves this purpose and is improved in every release by adding new features. However, under many circumstances, these new features have not been fully utilized by customers or are being used in incorrect ways. Customers often use scripts and settings from one release to the next, based on earlier experience. These settings and methods may have been superseded, or produce statistics that no longer give the most effective optimizer results.

The Optimizer Statistics Advisor in Oracle Database 12c Release 2 uses rules consistent with the current release to recommend changes to the way statistics are being gathered.

The advisor has a set of rules or recommended practices that are compared against the current statistics to generate findings. The rules are applied at the system, operation, or object level, such as whether the Automatic Gather Statistics jobs are scheduled, the statistics gathering procedures are using default parameters, and statistics are consistent across related objects. These rules check on issues related to the gathering of statistics - the schedules, parameters, and errors related to the automatic statistics gathering jobs. The rules include a variety of object-related issues, including whether incremental mode setting is efficient.

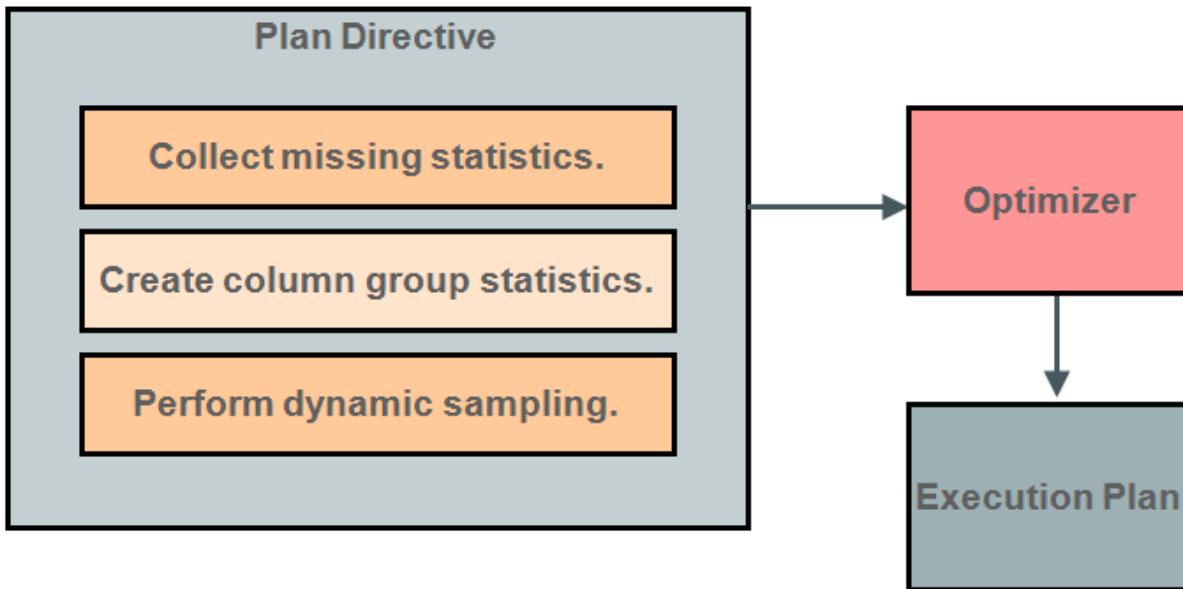
An Optimizer Statistics Advisor task can be executed with PL/SQL calls. Each task must be provided a unique task name. The definition of the CREATE ADVISEDOR_TASK() function parameters are:

- **TASK_NAME:** Name of the Statistics Advisor task
- **TIME_LIMIT:** The maximum duration the task can run

A filter list can be applied to the task to limit the scope of an advisor task using inclusion or exclusion lists for a user-specified set of rules, schemas, or operations. System rules are always checked. For example, you can configure an advisor task to include only recommendations for the SH schema. Also, you could exclude all violations of the rule for stale statistics.

You can create filters with the following DBMS_STATS procedures either individually or in combination, CONFIGURE_ADVISOR_OBJ_FILTER, CONFIGURE_ADVISOR_RULE_FILTER and CONFIGURE_ADVISOR_OPR_FILTER. An Optimizer Statistics Advisor Report can be generated with PL/SQL calls. The REPORT_ADVISOR_TASK function produces a report in text, HTML, or XML format. The IMPLEMENT_ADVISOR_TASK implements the recommendations of the task based on the filters in place. An additional parameter, LEVEL, can be set to either TYPICAL or ALL. TYPICAL is the default. ALL ignores the filters.

SQL Plan Directives



ORACLE®

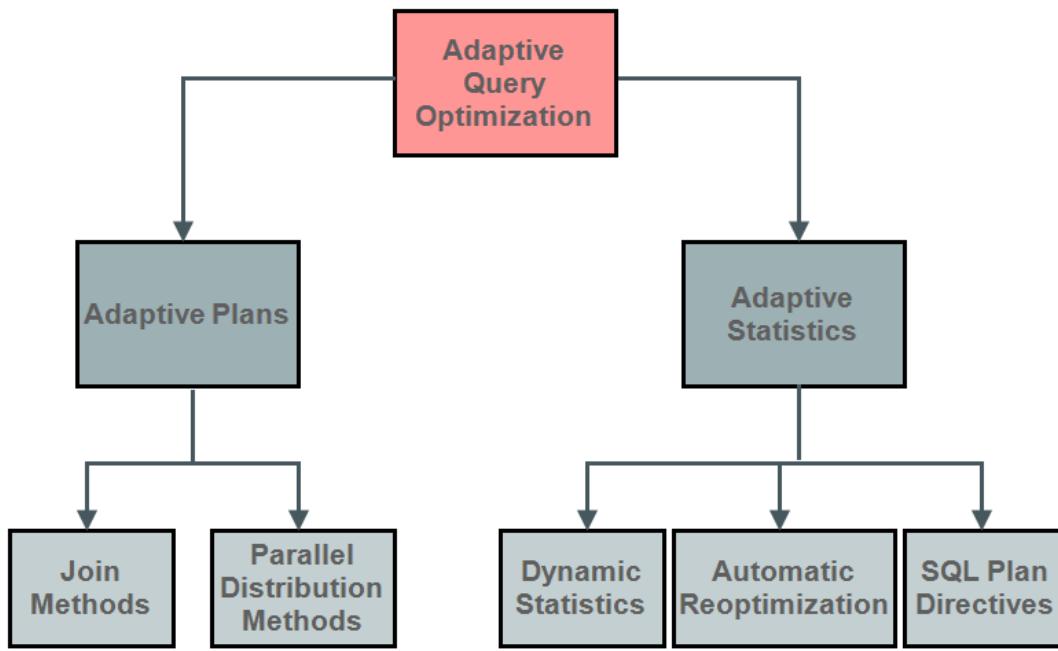
The Oracle Database 12c server can use a SQL plan directive, which is additional information and instructions that the optimizer can use to generate a more optimal plan. For example, a SQL plan directive might instruct the optimizer to collect missing statistics, create column group statistics, or perform dynamic sampling. During SQL compilation or execution, the database analyzes the query expressions that are missing statistics or that misestimate optimizer cardinality to create SQL plan directives. When the optimizer generates an execution plan, the directives give the optimizer additional information about objects that are referenced in the plan.

SQL plan directives are not tied to a specific SQL statement or SQL ID. The optimizer can use SQL plan directives for SQL statements that are nearly identical because SQL plan directives are defined on a query expression. For example, directives can help the optimizer with queries that use similar patterns, such as web-based queries that are the same except for a select list item. The database stores SQL plan directives persistently in the SYSAUX tablespace. When generating an execution plan, the optimizer can use SQL plan directives to obtain more information about the objects that are accessed in the plan.

Directives are automatically maintained, created as needed, and purged if not used after a year.

Directives can be monitored in DBA_SQL_PLAN_DIR_OBJECTS. SQL plan directives improve plan accuracy by persisting both compilation and execution statistics in the SYSAUX tablespace, allowing them to be used by multiple SQL statements.

Adaptive Execution Plans



ORACLE®

The Adaptive Execution Plans feature enables the optimizer to automatically adapt a poorly performing execution plan at run time and prevent a poor plan from being chosen on subsequent executions. The optimizer instruments its chosen plan so that at run time, it can be detected if the optimizer's estimates are not optimal. Then the plan can be automatically adapted to the actual conditions. An adaptive plan is a plan that changes after optimization when optimizer estimates prove inaccurate.

The optimizer can adapt plans based on statistics that are collected during statement execution. All adaptive mechanisms can execute a plan that differs from the plan that was originally determined during hard parse. This improves the ability of the query-processing engine (compilation and execution) to generate better execution plans.

The two Adaptive Execution Plan techniques are:

- **Dynamic plans:** A dynamic plan chooses among subplans during statement execution. For dynamic plans, the optimizer must decide which subplans to include in a dynamic plan, which statistics to collect to choose a subplan, and thresholds for this choice.
- **Re-optimization:** In contrast, re-optimization changes a plan for executions after the current execution. For re-optimization, the optimizer must decide which statistics to collect at which points in a plan and when re-optimization is feasible.

Note: OPTIMIZER_ADAPTIVE_REPORTING_ONLY controls reporting-only mode for adaptive optimizations. When set to TRUE, adaptive optimizations run in reporting-only mode where the information required for an adaptive optimization is gathered, but no action is taken to change the plan.

In Oracle Database 12c Release 2, the optimizer can pick the best-performing plan during any execution of the statement, not just the first execution. If the underlying data changes, or if queries re-execute with different input data, then the optimizer can adapt its plan to match the statistics gathered in the current execution. The continuous adaptive query plan adapts for every execution of the same cursor instead of only once.

SQL Tuning Advisor

THESE eKIT MATERIALS ARE FOR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



SQL Tuning Advisor

Comprehensive SQL tuning

- Detect stale or missing statistics
- Tune SQL plan (SQL profile)
- Add missing index
- Restructure SQL

ORACLE®

The SQL Tuning Advisor is the primary driver of the tuning process. It performs several types of analyses:

- **Statistics Analysis:** Checks each query object for missing or stale statistics, and makes recommendations to gather relevant statistics
- **SQL Profiling:** The optimizer verifies its own estimates and collects auxiliary information to remove estimation errors. It builds a SQL profile using the auxiliary information and makes a recommendation to create it. When a SQL profile is created, it enables the query optimizer to generate a well-tuned plan.
- **Access Path Analysis:** New indexes are considered if they significantly improve access to each table in the query. When appropriate, recommendations to create such objects are made.
- **SQL Structure Analysis:** SQL statements that use bad plans are identified and relevant suggestions are made to restructure them. The suggested changes can be syntactic as well as semantic.

The SQL Tuning Advisor considers each SQL statement included in the advisor task independently. Creating a new index may help a query, but may hurt the response time of DML. So, a recommended index or other object should be checked with the SQL Access Advisor over a workload (a set of SQL statements) to determine whether there is a net gain in performance.

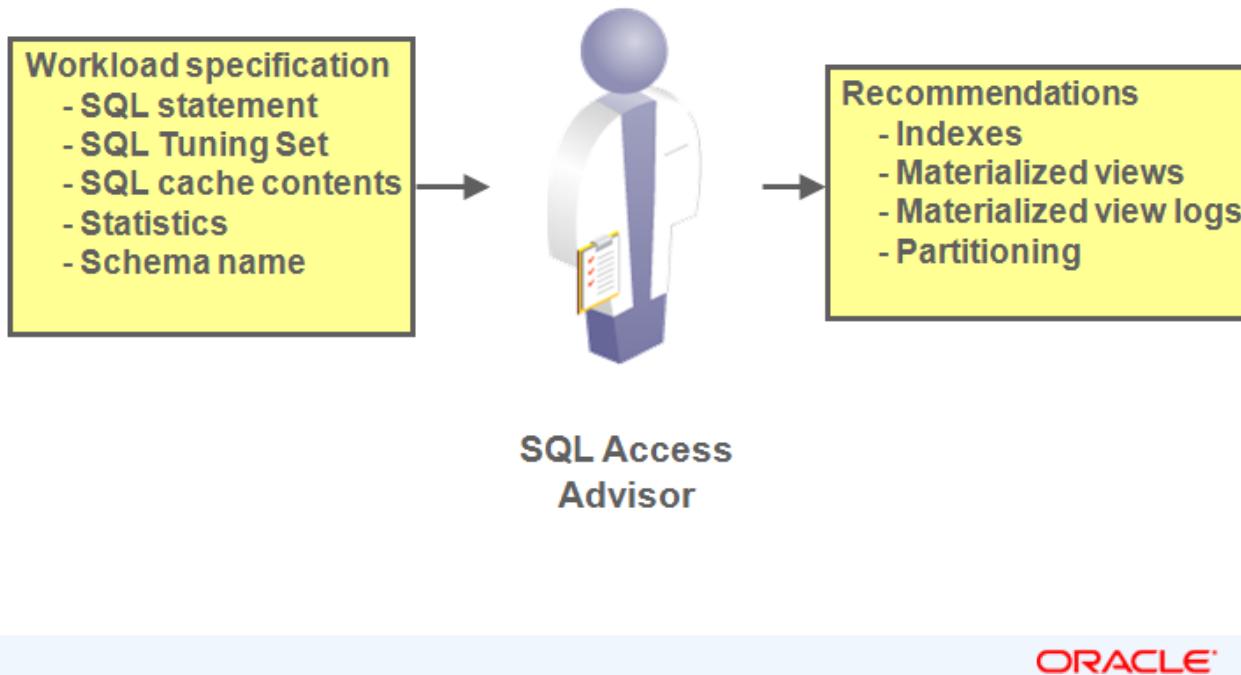
The SQL Tuning Advisor runs automatically every night as the Automatic SQL Tuning Task. There may be times when a SQL statement needs immediate tuning action. You can use the SQL Tuning Advisor to analyze SQL statements and obtain performance recommendations at any time. Typically, you run this advisor as an ADDM performance-finding action.

Additionally, you can run the SQL Tuning Advisor when you want to analyze the top SQL statements consuming the most CPU time, I/O, and memory.

Even though you can submit multiple statements to be analyzed in a single task, each statement is analyzed independently. To obtain tuning recommendations that consider overall performance of a set of SQL, use the SQL Access Advisor.

SQL Access Advisor

THESE eKIT MATERIALS ARE FOR YOUR USE ONLY. COPYING OR DISTRIBUTION IS PROHIBITED.



ORACLE®

The SQL Access Advisor can recommend the proper set of materialized views, materialized view logs, partitioning, and indexes for a given workload. Understanding and using these structures is essential when optimizing SQL because they can result in significant performance improvements in data retrieval.

The SQL Access Advisor recommends bitmap, function-based, and B-tree indexes. A bitmap index offers a reduced response time for many types of ad hoc queries and reduced storage requirements compared to other indexing techniques. B-tree indexes are most commonly used in a data warehouse to index unique or near-unique keys.

Another component of the SQL Access Advisor also recommends how to optimize materialized views so that they can be fast refreshable and take advantage of general query rewrite.

Note: For more information about materialized views and query rewrite, see the *Oracle Database Performance Tuning Guide*.

SQL Performance Analyzer



Predicts impact of system changes

Builds different versions of SQL workload performance

Executes SQL serially

Analyzes performance differences

Offers fine-grained performance analysis on individual SQL

ORACLE®

Oracle Database includes SQL Performance Analyzer, which gives you an exact and accurate assessment of the impact of change on the SQL statements that make up the workload. SQL Performance Analyzer helps you forecast the impact of a potential change on the performance of a SQL query workload. This capability provides you with detailed information about the performance of SQL statements, such as before-and-after execution statistics, and statements with performance improvement or degradation. This enables you (for example) to make changes in a test environment to determine whether the workload performance will be improved through a database upgrade.

SQL Performance Analyzer includes the following capabilities:

- Helps predict the impact of system changes on SQL workload response time
- Builds different versions of SQL workload performance (that is, SQL execution plans and execution statistics)
- Executes SQL serially (concurrency not honored)
- Analyzes performance differences
- Offers fine-grained performance analysis on individual SQL
- Is integrated with SQL Tuning Advisor to tune regressions

Use Cases

SQL Performance Analyzer can be used to predict and prevent potential performance problems for any database environment change that affects the structure of the SQL execution plans. The changes can include (but are not limited to) any of the following:

- Database upgrades
- Implementation of tuning recommendations
- Schema changes
- Statistics gathering
- Database parameter changes
- OS and hardware changes

You can use SQL Performance Analyzer to predict SQL performance changes that result from changes for even the most complex environments. As applications evolve through the development life cycle, database application developers can test changes to schemas, database objects, and rewritten applications to mitigate any potential performance impact.

SQL Performance Analyzer also enables the comparison of SQL performance statistics.

You can access SQL Performance Analyzer through Enterprise Manager or by using the `DBMS_SQLPA` package.

For details about the `DBMS_SQLPA` package, see the *Oracle Database PL/SQL Packages and Types Reference Guide*.

Summary for Lesson 12

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

In this lesson, you should have learned to:

- Describe the SQL tuning methodology
- Manage optimizer statistics
- Use the SQL Tuning Advisor to identify and tune SQL statements that are using the most resources
- Use the SQL Access Advisor to tune a workload



ORACLE®

Practice 12 Overview

- 12-1: Using SQL Tuning Advisor
- 12-2: Using Optimizer Statistics Advisor



Practice 12-1 Using SQL Tuning Advisor

Overview

In this practice, you optimize the performance of a costly SQL statement by using the SQL Tuning Advisor through EM Express.

Note: All advisors are also available through EM Cloud Control.

Assumptions

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

Tasks

Run a SQL Load

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. Execute the `$HOME/labs/PERF_setup_tuning.sh` shell script. This script recreates PDB1 and PDB2. In PDB1, it creates a user called `admin_pdb1`, a tablespace called `TBS_APP`, and a schema called `oe` in the `TBS_APP` tablespace. It does the same thing in PDB2, except it creates a user called `admin_pdb2`. Wait for the setup script to finish. It may take a couple of minutes to run. You can ignore any error messages because they are expected.

```
$ $HOME/labs/PERF_setup_tuning.sh

...
$
```

3. Start an application workload in PDB1 and PDB2. The `PERF_loop.sh` script runs a SQL script named `PERF_loop.sql` eight times in PDB1 as the `oe` user. It then runs the same SQL script eight times in PDB2 as the `SYSTEM` user. You can move on to the next step while the code is running.

```
$ $HOME/labs/PERF_loop.sh

...
$
```

Tune the SQL in EM Express Based on Statistics

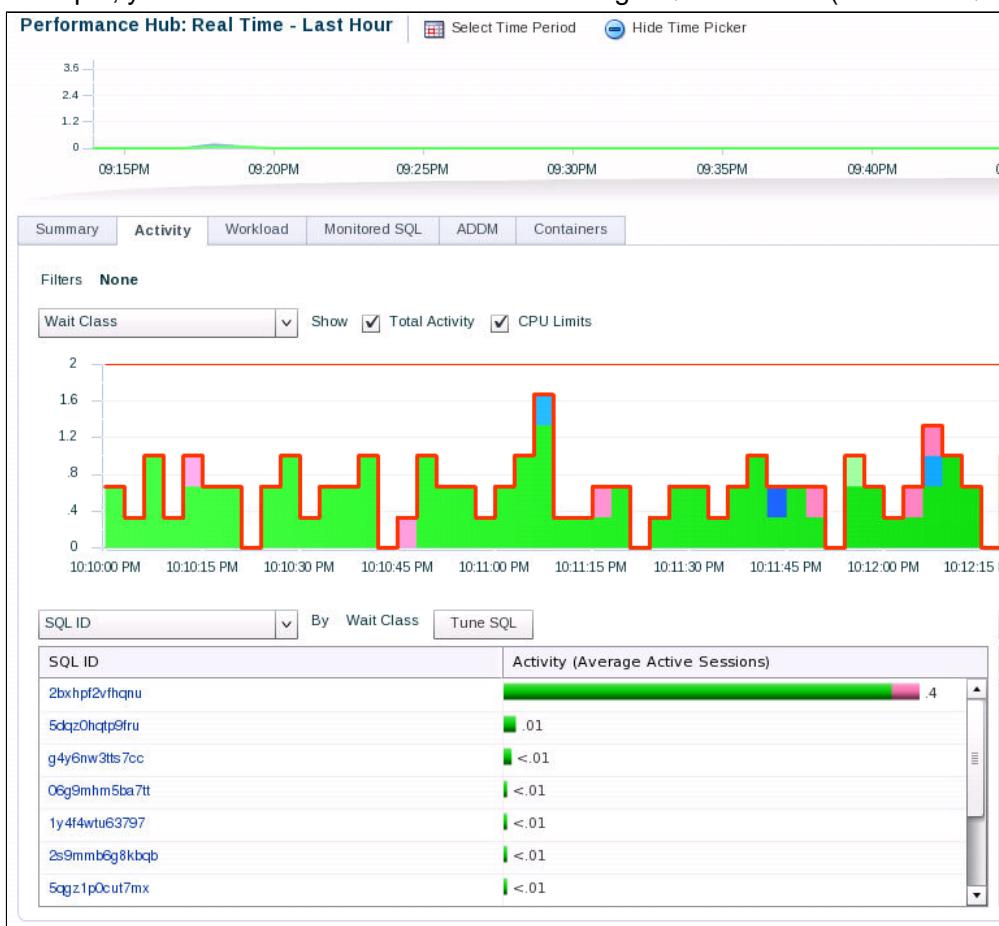
In this section, you review, not implement, the first recommendation of SQL Tuning Advisor, which is based on statistics.

1. Connect the client browser to the CDB in EM Express.

- Open a Firefox web browser, and enter the URL <https://localhost:5500/em>.
- On the Login page, enter the user name **sys** and the password as specified in [Appendix D - Product-Specific Credentials](#). Leave the Container Name box empty, select the **as sysdba** check box, and click **Login**.

2. Select **Performance**, and then **Performance Hub**.

3. Click the **Activity** tab. The current SQL executions are listed in the table at the bottom of the page. In this example, you can see that there is one consuming SQL execution (the first SQL ID in the list).



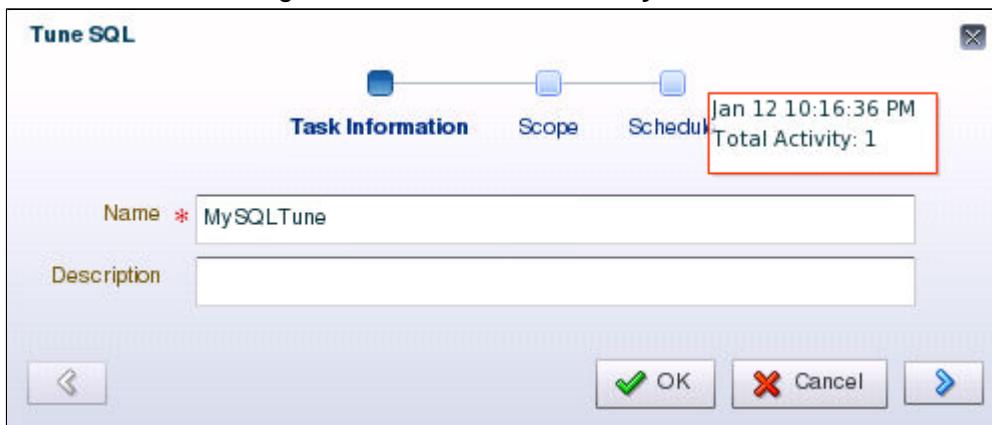
4. Position your cursor over the SQL ID. The following code should appear. If your result looks different, wait a moment, and refresh EM Express.

```
SELECT /*+ monitor USE_NL(d l pi i)          FULL(d)
          FULL(l) FULL(pi) FULL(i) */...
```

5. Click the SQL ID, and then click the **Execution Statistics** tab. The Plan Details area at the bottom of the page shows you the current plan for executing the SQL.

Operation	Object
SELECT STATEMENT	
HASH GROUP BY	
HASH JOIN	
VIEW	VW_GBF_16
HASH GROUP BY	
NESTED LOOPS	
TABLE ACCESS FULL	PRODUCT_INFORMATION
TABLE ACCESS FULL	INVENTORIES
NESTED LOOPS	
TABLE ACCESS FULL	ORDERS
TABLE ACCESS FULL	ORDER_ITEMS

6. To return to where you were last, select **Performance**, and then **Performance Hub**. Click the **Activity** tab.
 7. In the Activity column, click the bar to select the SQL statement to tune, and click the **Tune SQL** button to launch the SQL Tuning Advisor.
 8. Question: What may the SQL Tuning Advisor suggest?
 Answer: It can suggest indexing columns, SQL profiles implementation, restructuring the SQL statement, and collecting missing or stale object statistics.
 9. In the Tune SQL dialog box, enter the task name **MySQLTune**, and click **OK**.



10. While the analysis task is completing, you are automatically brought to the SQL Tuning Advisor page, which has two tabs:
- The Automatic tab lists the automatic tasks executed every night.
 - The Manual tab lists the manually created SQL tuning tasks.

11. Click the **Manual** tab. The task you just created is listed. You may need to wait a moment for it to complete its processing.

The screenshot shows the SQL Tuning Advisor interface. At the top, there are tabs for 'Automatic Runs' and 'Show All'. Below that, a message states: 'The SQL Tuning Advisor analyzes individual SQL statements, and suggests indexes, SQL profiles, restructured SQL, and statistics to improve their performance. It runs automatically during the maintenance window or can be invoked manually on selected SQL statements.' A 'Configuration' link is also present. The main area has a heading 'This table shows a list of manually created SQL tuning tasks. You can create new SQL tuning tasks from the Performance Hub by running the SQL Tuning advisor on high-load SQL statements.' Below this is a table with columns: Actions, Status, Name, Description, User, Start Time, Duration, and Expires In (days). One row is visible: MySQLTune (Status: Active), User: SYS, Start Time: Thu Jan 12, 2017 10:20:34 PM, Duration: 10m, Expires In (days): 30. There are buttons for 'View Result' and 'Drop'.

12. Click your task to read the recommendations.
13. In the SQL Text area, you can view the full SQL statement to be tuned. Scroll down to view all of it.

The screenshot shows the SQL Text area with a large block of SQL code. The code is a SELECT statement with various joins and conditions. It includes clauses like /*+ monitor USE_NL(d l pi i) FULL(d) FULL(l) FULL(pi) FULL(i) */ and WHERE clauses involving oe.orders, oe.order_items, oe.product_information, and oe.inventories tables.

```

SELECT /*+ monitor USE_NL(d l pi i)
      FULL(d) FULL(l) FULL(pi) FULL(i) */
      l.order_id, SUM(unit_price * quantity) amount
   FROM  oe.orders d , oe.order_items l,
         oe.product_information pi, oe.inventories i
  WHERE d.order_id = l.order_id
    AND pi.product_id = l.product_id
    AND pi.product_id = i.product_id
    AND d.order_date < to_DATE('10-12-2021','DD-MM-YYYY')
    AND d.order_total BETWEEN 1394 AND 100000
    AND l.quantity between 10 AND 300
  
```

14. In the Select Recommendation area, notice that there are two recommendations: Statistics and SQL Profile. For the Statistics recommendation, SQL Tuning Advisor's findings say that Optimizer statistics are not up-to-date for objects referenced in the SQL statement. It identifies OE tables in the SQL statement that are not analyzed.

The screenshot shows the 'Select Recommendation' section. It says 'Only one recommendation should be implemented.' There are three buttons: 'View Details', 'Implement', and 'Validate with SPA' (with a checked checkbox). The table below lists recommendations:

Type	Findings
Statistics	Optimizer statistics are not up-to-date for objects referenced in the SQL statement. Table "OE"."ORDER_ITEMS" was not analyzed. Table "OE"."INVENTORIES" was not analyzed. Table "OE"."PRODUCT_INFORMATION" was not analyzed. Table "OE"."ORDERS" was not analyzed.
SQL Profile	A potentially better execution plan was found for this statement.

15. Question: How do tables get statistics collected automatically?
 Answer: There is an automated task that automatically gathers Optimizer statistics every night. You can configure the settings that are used for Optimizer statistics gathering.
16. In the Select Recommendations area, click the **Statistics** link.

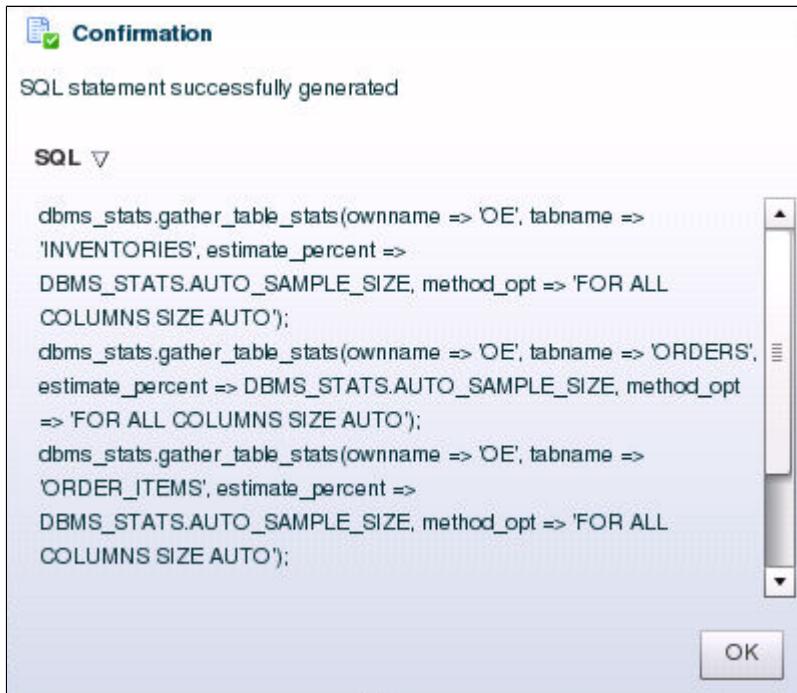
17. On the Recommendation Details page, view the list of recommendations, and then click the **Implement** button at the top of the page. Notice that the Optimizer recommends gathering statistics for the tables in your SQL statement.

The screenshot shows the Oracle Database 12c Performance section. Under 'Recommendation Details', it lists several recommendations for gathering statistics on tables: PRODUCT_INFORMATION, INVENTORIES, ORDER_ITEMS, and ORDERS. Below this, the 'Compare Explain Plans' section shows the execution plan for a query involving these tables. The plan details various operations like SELECT STATEMENT, HASH GROUP BY, and HASH JOIN, along with their estimated rows and bytes.

18. In the Gather Statistics dialog box, click **Show SQL**.



19. View the SQL package and procedure that would gather the statistics, and click **OK**.

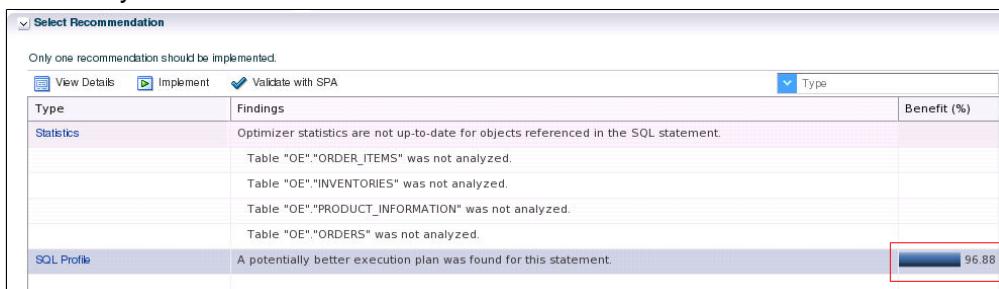


20. In the Gather Statistics dialog box, click **Cancel** because you will ask the Optimizer Advisor in the next practice to help you implement the statistics collection in the best way.

Tune the SQL in EM Express Based on a SQL Profile

In this section, you implement the second recommendation, which suggests the usage of a SQL profile. This option provides a better execution plan.

1. Select **Performance**, and then **SQL Tuning Advisor**.
2. Click the **Manual** tab, if needed.
3. Click **MySQLTune** (or the name you gave your tuning task).
4. In the Select Recommendation section, at the bottom of the Benefit column, view the benefit percentage value for using the SQL Profile. In this example, the SQL profile would increase performance by almost 97%. Your value may be different.



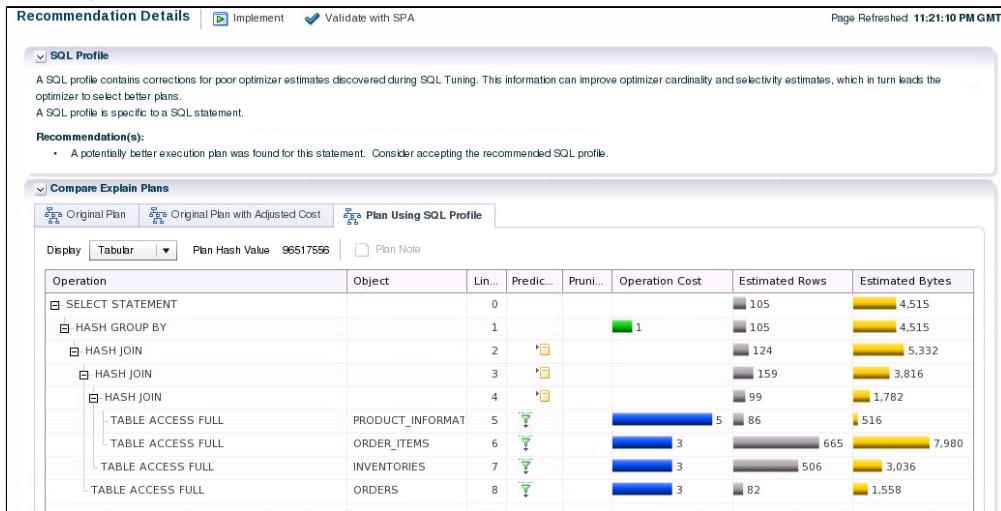
5. In the Select Recommendation section, click the **SQL Profile** link.

Type	Findings
Statistics	Optimizer statistics are not up-to-date for objects referenced in the SQL statement. Table "OE"."ORDER_ITEMS" was not analyzed. Table "OE"."INVENTORIES" was not analyzed. Table "OE"."PRODUCT_INFORMATION" was not analyzed. Table "OE"."ORDERS" was not analyzed.
SQL Profile	A potentially better execution plan was found for this statement.

6. On the Original Plan tab, which is displayed by default, view the SQL execution plan. You've seen this plan earlier in the practice.

Operation	Object	Line...	Predic...	Pruning	Operation Cost	Estimated Rows	Estimated Bytes
SELECT STATEMENT		0				453	53K
HASH GROUP BY		1		1		453	53K
HASH JOIN		2	1	1		453	53K
VIEW	VW_GBF_16	3				312	8,112
HASH GROUP BY		4	1			312	16K
NESTED LOOPS		5		117		312	16K
TABLE ACCESS FULL	PRODUCT_INFORMATION	6	1	5		86	2,236
TABLE ACCESS FULL	INVENTORIES	7	1	1		4	104
NESTED LOOPS		8		112		416	38K
TABLE ACCESS FULL	ORDERS	9	1	3		82	3,198
TABLE ACCESS FULL	ORDER_ITEMS	10	1	1		5	260

7. Click the **Plan Using SQL Profile** tab and view its SQL execution plan. Notice the differences between it and the original plan.



8. At the top of the page, click the **Implement** button.
 9. In the Create SQL Profile dialog box, click **Show SQL**. A Confirmation dialog box shows you the generated SQL statement. Click **OK**.

```
dbms_sqltune.accept_sql_profile(task_name => 'MySQLTune', task_owner
=>'SYS', replace => TRUE);
```

10. In the Create SQL Profile dialog box, click **OK** to implement the new profile. The SQL profile is created.
 11. In the Confirmation dialog box, click **OK**.

Rerun the SQL Script and Verify the Performance Benefit

In this section, you re-execute the `PERF_loop.sh` script and verify that the SQL tuning you just implemented made the query less consuming in the database.

1. Return to the terminal window.
2. If the previous script is still running, press **Ctrl+c** to stop the activity.
3. Start SQL*Plus and connect to the CDB root as the `SYS` user with the `SYSDBA` privilege.

```
$ sqlplus / AS SYSDBA
...
SQL>
```

4. When testing SQL, it is a good idea to periodically flush the shared pool entries to remove older execution plans.

```
SQL> ALTER SYSTEM FLUSH SHARED_POOL;
System altered.
SQL>
```

- Remove any blocks on the tables (selected in the query) from the buffer cache.

```
SQL> ALTER SYSTEM FLUSH BUFFER_CACHE;

System altered.
SQL>
```

- Exit SQL*Plus.

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

- Run the application workload again in PDB1 and PDB2. The `PERF_loop.sh` script runs a SQL script named `PERF_loop.sql` eight times in PDB1 as the `oe` user. It then runs the same SQL script eight times in PDB2 as the `SYSTEM` user.

```
$ $HOME/labs/PERF_loop.sh
...
$
```

- Return to EM Express.
- If a Warning dialog box is displayed stating that a particular SQL ID is no longer from Cursor Cache, click **OK**.
- Select **Performance**, and then **Performance Hub**.
- Click the **Activity** tab.
- At the bottom of the page, note the value in the Activity column.



- Question: How does the value in the Activity column now compare to the value in the Activity column prior to SQL tuning? Is there a performance benefit to the SQL tuning that you just did?
Answer: In this example, the average active sessions value went down from .4 to .16, so yes, there is a performance benefit from tuning the SQL. Your values may differ.
- Click the link for the SQL ID. The Summary tab is displayed for the SQL ID.
- Click the **Execution Statistics** tab. In the Plan Details area, notice that the plan now used is the plan that uses the SQL Profile (refer back to step 7 in the previous section in this practice).

Plan Details	
Display	Tabular
Plan Hash Value	96517556
	 Plan Note
Operation	Object
└ HASH JOIN	
└ HASH JOIN	
└ HASH JOIN	
└ TABLE ACCESS FULL	PRODUCT_INFORMATIC
└ TABLE ACCESS FULL	ORDER_ITEMS
└ TABLE ACCESS FULL	INVENTORIES
└ TABLE ACCESS FULL	ORDERS

16. Click **Log Out** to exit EM Express and close the browser window.
17. In the terminal window, press **Ctrl+c** to stop the activity. Close the terminal window.

Practice 12-2 Using Optimizer Statistics Advisor

Overview

In this practice, you learn how to improve optimizer statistics collection quality with the new Oracle Database 12.2 Optimizer Statistics Advisor.

The advisor task runs automatically in the maintenance window, but you can also run it on demand. If the advisor makes findings and then recommendations, then in some cases you can run system-generated scripts to implement them. Optimizer statistics play a significant part in determining the execution plan for queries. Therefore, it is critical for the optimizer to gather and maintain accurate and up-to-date statistics. All findings are derived from rules, but not all rules generate findings.

Assumptions

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

Tasks

Start an Application Workload in Window 1

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as Window 1.

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

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

2. Execute the `$HOME/labs/PERF_setup_tuning.sh` shell script. This script recreates PDB1 and PDB2. In PDB1, it creates a user called `admin_pdb1`, a tablespace called `TBS_APP`, and a schema called `oe` in the `TBS_APP` tablespace. It does the same thing in PDB2, except it creates a user called `admin_pdb2`. Wait for the setup script to finish. It may take a couple of minutes to run. You can ignore any error messages because they are expected.

```
$ $HOME/labs/PERF_setup_tuning.sh

...
$
```

3. Start an application workload in PDB1 and PDB2.

```
$ $HOME/labs/PERF_loop.sh

...
$
```

Use the Optimizer Statistics Advisor

In this section, you create an object filter for an Optimizer Statistics Advisor task, create and execute the task, generate a report with recommendations, and then implement those recommendations.

1. Open another new terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL . This window will be referred to as window 2.

```
$ . 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. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYS/<password>@PDB1 AS SYSDBA

...
SQL>
```

3. Execute the following procedure, which is an object filter for an Optimizer Advisor Task. This filter disables statistics collection recommendations for all objects except those in the OE schema. The reason you want to do this is because according to [Practice 12-1 Using SQL Tuning Advisor](#), the OE tables were included in queries that required tuning.

Note: If you prefer, you can enter @\$HOME/labs/OPTADV_1.sql to run a SQL script instead of typing the following code.

```
SQL> CREATE OR REPLACE PROCEDURE sh_obj_filter(p_tname IN VARCHAR2) IS v_retc
CLOB;
```

```
2   BEGIN
3     v_retc := DBMS_STATS.CONFIGURE_ADVISOR_OBJ_FILTER
        (p_tname,'EXECUTE',NULL,NULL,NULL,'DISABLE');
4     v_retc := DBMS_STATS.CONFIGURE_ADVISOR_OBJ_FILTER
        (p_tname,'EXECUTE',NULL,'OE',NULL,'ENABLE');
5   END;
6 /
```

```
Procedure created.
```

```
SQL>
```

4. Create and execute an advisor task named my_task by issuing the following code.

Note: If you prefer, you can enter @\$HOME/labs/OPTADV_2.sql to run a SQL script instead of typing the following code. The script takes about ten seconds to run.

```
SQL> DECLARE
  2   v_tname  VARCHAR2(128) := 'my_task';
  3   v_ename  VARCHAR2(128) := NULL;
  4   v_report  CLOB := null;
  5   v_script  CLOB := null;
  6   v_implementaion_result CLOB;
  7   BEGIN
  8     v_tname := DBMS_STATS.CREATE_ADVISOR_TASK(v_tname);
  9     sh_obj_filter(v_tname);
 10    v_ename := DBMS_STATS.EXECUTE_ADVISOR_TASK(v_tname);
 11  END;
 12 /
```

PL/SQL procedure successfully completed.
SQL>

5. Verify that the procedure completed successfully.

- a. Query the USER_ADVISOR_TASKS view. The results below are formatted for easier viewing.

```
SQL> SELECT advisor_name, execution_type, last_execution, status
  2   FROM user_advisor_tasks
  3  WHERE task_name = 'MY_TASK';

ADVISOR_NAME      EXECUTION_TYPE LAST_EXECUTION STATUS
-----  -----  -----  -----
Statistics Advisor STATISTICS      EXEC_11      COMPLETED
SQL>
```

- b. Query the USER_ADVISOR_EXECUTIONS view. The results below are formatted for easier viewing. Your dates will be different than those shown below.

```
SQL> SELECT task_name, execution_name, execution_end, execution_type AS
  type, status
  2   FROM user_advisor_executions;

TASK_NAME      EXECUTION_NAME EXECUTION_END      TYPE      STATUS
-----  -----  -----  -----
AUTO_STATS_ADVISOR_TASK EXEC_1      16-MAY-16      STATISTICS
COMPLETED
MY_TASK        EXEC_11      09-NOV-16      STATISTICS
COMPLETED
SQL>
```

6. Generate a report.

- a. Run the following procedure.

Note: If you prefer, you can enter @\$HOME/labs/OPTADV_3.sql to run a SQL script instead of typing the following code.

```
SQL> VAR b_report CLOB
SQL> DECLARE
  2  v_tname VARCHAR2(32767);
  3  BEGIN
  4  v_tname := 'my_task';
  5  :b_report := dbms_stats.report_advisor_task(v_tname, type => 'TEXT',
section=>'ALL', level=>'ALL');
  6  END;
  7  /

PL/SQL procedure successfully completed.

SQL>
```

- b. Run the following procedure.

Note: If you prefer, you can enter @\$HOME/labs/OPTADV_4.sql to run a SQL script instead of typing the following code.

```
SQL> DECLARE
  2  v_len NUMBER(10);
  3  v_offset NUMBER(10) :=1;
  4  v_amount NUMBER(10) :=10000;
  5  BEGIN
  6  v_len := DBMS_LOB.getlength(:b_report);
  7  WHILE (v_offset < v_len)
  8  LOOP
  9  DBMS_OUTPUT.PUT_LINE(DBMS_LOB.SUBSTR(:b_report,v_amount,v_offset));
10 v_offset := v_offset + v_amount;
11 END LOOP;
12 END;
13 /

PL/SQL procedure successfully completed.

SQL>
```

7. Display the findings and recommendations for each of them.

Note: If you prefer, you can enter @\$HOME/labs/OPTADV_5.sql to run a SQL script instead of typing the following code.

```

SQL> SELECT f.finding_id, f.message, r.benefit_type
  2  FROM user_advisor_findings f, user_advisor_recommendations r
  3  WHERE  f.finding_id = r.finding_id AND f.task_name = 'MY_TASK' AND
f.execution_name = 'EXEC_11';

FINDING_ID
-----
MESSAGE
-----
BENEFIT_TYPE
-----
      1
There are 10 object(s) with no statistics.
Set parameter job_queue_processes to 1 or higher.
      1
There are 10 object(s) with no statistics.
Gather Statistics on those objects with no statistics.

SQL>

```

8. Generate the script before a possible implementation.

a. Execute the following SQL.

Note: If you prefer, you can enter @\$HOME/labs/OPTADV_6.sql to run a SQL script instead of typing the following code.

```

SQL> SET SERVEROUTPUT ON
SQL> VARIABLE b_script CLOB
SQL> DECLARE
  2   v_tname VARCHAR2(32767);
  3   BEGIN
  4     v_tname := 'my_task';
  5     :b_script := DBMS_STATS.SCRIPT_ADVISOR_TASK(v_tname);
  6   END;
  7 / 

PL/SQL procedure successfully completed.
SQL>

```

b. Execute the following SQL.

Note: If you prefer, you can enter @\$HOME/labs/OPTADV_7.sql to run a SQL script instead of typing the following code.

```

SQL> DECLARE
  2   v_len NUMBER(10);
  3   v_offset NUMBER(10) :=1;
  4   v_amount NUMBER(10) :=10000;
  5   BEGIN
  6     v_len := DBMS_LOB.getlength(:b_report);
  7     WHILE (v_offset < v_len)
  8     LOOP

```

```
9  DBMS_OUTPUT.PUT_LINE(DBMS_LOB.SUBSTR(:b_script, v_amount, v_offset));
10 v_offset := v_offset + v_amount;
11 END LOOP;
12 END;
13 /

-- Script generated for the recommendations from execution EXEC_11
-- in the statistics advisor task MY_TASK
-- Script version 12.2
-- No scripts will be provided for the rule USEAUTOJOB. Please check the
report for more details.
-- No scripts will be provided for the rule COMPLETEAUTOJOB. Please check
the report for more details.
-- No scripts will be provided for the rule MAINTAINSTATSHISTORY. Please
check the report for more details.
-- No scripts will be provided for the rule TURNONSQLPLANDIRECTIVE. Please
check the report for more details.
-- No scripts will be provided for the rule AVOIDSETPROCEDURES. Please check
the report for more details.
-- No scripts will be provided for the rule USEDEFAULTPARAMS. Please check
the report for more details.
-- No scripts will be provided for the rule USEGATHERSCHEMASTATS. Please
check the report for more details.
-- No scripts will be provided for the rule
AVOIDINEFFICIENTSTATSOPRSEQ. Please check the report for more details.
-- No scripts will be provided for the rule
AVOIDUNNECESSARYSTATSCOLLECTION. Please check the report for more details.
-- No scripts will be provided for the rule GATHERSTATSAFTERBULKDML. Please
check the report for more details.
-- No scripts will be provided for the rule AVOIDDROPRECREATE. Please check
the report for more details.
-- No scripts will be provided for the rule AVOIDOUTOFRANGE. Please check
the report for more details.
-- No scripts will be provided for the rule AVOIDANALYZETABLE. Please check
the report for more details.
-- No scripts will be provided for the rule USEAUTOJOB. Please check the
report for more details.
-- No scripts will be provided for the rule COMPLETEAUTOJOB. Please check
the report for more details.
-- No scripts will be provided for the rule MAINTAINSTATSHISTORY. Please
check the report for more details.
-- No scripts will be provided for the rule TURNONSQLPLANDIRECTIVE. Please
check the report for more details.
...
--

Scripts for rule USECONCURRENT
-- Rule Description: Use Concurrent preference for Statistics Collection
-- No scripts will be provided for the rule USEAUTOJOB. Please check the
report for more details.
-- No scripts will be provided for the rule COMPLETEAUTOJOB. Please check
the report for more details.
-- No scripts will be provided for the rule MAINTAINSTATSHISTORY. Please
check the report for more details.
...
```

```
-- Scripts for rule USEDEFAULTOBJECTPREFERENCE
--
Rule Description: Use Default Object Preference for statistics collection
-- Setting object-level preferences to default values
-- setting CASCADE to default value for object level preference
-- setting ESTIMATE_PERCENT to default value for object level preference
-- setting METHOD_OPT to default value for object level preference
-- setting GRANULARITY to default value for object level preference
-- setting NO_INVALIDATE to default value for object level preference
-- Scripts for rule USEINCREMENTAL
...
-- Scripts for rule MAINTAINSTATSCONSISTENCY
--
Rule Description: Statistics of dependent objects should be consistent
-- Gather statistics for those objects that are missing or have no
statistics.
declare
    obj_filter_list dbms_stats.ObjectTab;
    obj_filter      dbms_stats.ObjectElem;
    obj_cnt         number := 0;
begin
    obj_filter_list := dbms_stats.ObjectTab();
    obj_filter.ownname := 'OE';
    obj_filter.objtype := 'TABLE';
    obj_filter.objname := 'CUSTOMER';
    obj_filter_list.extend();
    obj_cnt := obj_cnt + 1;
    obj_filter_list(obj_cnt) := obj_filter;
    obj_filter.ownname := 'OE';
    obj_filter.objtype := 'TABLE';
    obj_filter.objname := 'CUSTOMERS';
    obj_filter_list.extend();
    obj_cnt := obj_cnt + 1;
    obj_filter_list(obj_cnt) := obj_filter;
    obj_filter.ownname := 'OE';
    obj_filter.objtype := 'TABLE';
    obj_filter.objname := 'DATE_DIM';
    obj_filter_list.extend();
    obj_cnt := obj_cnt + 1;
    obj_filter_list(obj_cnt) := obj_filter;
    obj_filter.ownname := 'OE';
    obj_filter.objtype := 'TABLE';
    obj_filter.objname := 'INVENTORIES';
    obj_filter_list.extend();
    obj_cnt := obj_cnt + 1;
    obj_filter_list(obj_cnt) := obj_filter;
    obj_filter.ownname := 'OE';
    obj_filter.objtype := 'TABLE';
    obj_filter.objname := 'LINEORDER';
    obj_filter_list.extend();
    obj_cnt := obj_cnt + 1;
    obj_filter_list(obj_cnt) := obj_filter;
    obj_filter.ownname := 'OE';
    obj_filter.objtype := 'TABLE';
```

```
obj_filter.objname := 'ORDERS';
obj_filter_list.extend();
obj_cnt := obj_cnt + 1;
obj_filter_list(obj_cnt) := obj_filter;
obj_filter.ownname := 'OE';
obj_filter.objtype := 'TABLE';
obj_filter.objname := 'ORDER_ITEMS';
obj_filter_list.extend();
obj_cnt := obj_cnt + 1;
obj_filter_list(obj_cnt) := obj_filter;
obj_filter.ownname := 'OE';
obj_filter.objtype := 'TABLE';
obj_filter.objname := 'PART';
obj_filter_list.extend();
obj_cnt := obj_cnt + 1;
obj_filter_list(obj_cnt) := obj_filter;
obj_filter.ownname := 'OE';
obj_filter.objtype := 'TABLE';
obj_filter.objname := 'PRODUCT_INFORMATION';
obj_filter_list.extend();
obj_cnt := obj_cnt + 1;
obj_filter_list(obj_cnt) := obj_filter;
obj_filter.ownname := 'OE';
obj_filter.objtype := 'TABLE';
obj_filter.objname := 'SUPPLIER';
obj_filter_list.extend();
obj_cnt := obj_cnt + 1;
obj_filter_list(obj_cnt) := obj_filter;
dbms_stats.gather_database_stats(obj_filter_list=>obj_filter_list);
end;
/
PL/SQL procedure successfully completed.
SQL>
```

9. What does the `obj_filter_list` recommend to do in regard to the findings reported earlier?
Answer: The object filter list contains the names of the ten objects with no statistics that were reported in the previous step.
10. Question: What would you do if you agree with the recommendations?
Answer: You could either execute the generated SQL script or use the `DBMS_STATS.IMPLEMENT_ADVISOR_TASK`.
11. Execute the `DBMS_STATS.IMPLEMENT_ADVISOR_TASK` PL/SQL procedure. This procedure implements the actions recommended by the advisor based on results from a specified Optimizer Statistics Advisor execution.
Note: If you prefer, you can enter `@$HOME/labs/OPTADV_8.sql` to run a SQL script instead of typing the following code.

```
SQL> VARIABLE b_ret CLOB
SQL> DECLARE
  2  v_tname VARCHAR2(32767);
  3  BEGIN
  4  v_tname := 'my_task';
  5  :b_ret := DBMS_STATS.IMPLEMENT_ADVISOR_TASK(v_tname);
  6  END;
  7 /
```

PL/SQL procedure successfully completed.
SQL>

- Check that the statistics are collected for the ten objects. Columns like NUM_ROWS, EMPTY_BLOCKS, BLOCKS, and AVG_ROW_LEN have null values until the statistics are collected.

```
SQL> COL table_name FORMAT A20
SQL> SELECT table_name, num_rows, blocks, empty_blocks, avg_row_len,
last_analyzed
  2  FROM dba_tables WHERE owner='OE';
```

TABLE_NAME	NUM_ROWS	BLOCKS	EMPTY_BLOCKS	AVG_ROW_LEN	LAST_ANAL
CUSTOMERS	319	13	0	158	14-NOV-16
PRODUCT_INFORMATION	287	13	0	216	14-NOV-16
PART	1600000	20646	0	87	14-NOV-16
DATE_DIM	2556	43	0	100	14-NOV-16
INVENTORIES	636	5	0	11	14-NOV-16
ORDERS	105	5	0	38	14-NOV-16
ORDER_ITEMS	665	5	0	18	14-NOV-16
LINEORDER	3297032	47229	0	98	14-NOV-16
SUPPLIER	16000	260	0	102	14-NOV-16
CUSTOMER	240000	3846	0	107	14-NOV-16

10 rows selected.
SQL>

- Drop the advisor task.

```
SQL> exec DBMS_STATS.DROP_ADVISOR_TASK('my_task')
```

PL/SQL procedure successfully completed.
SQL>

- Exit SQL*Plus and close window 2.

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

15. In window 1, press **Ctrl+c** to stop the activity and close the window.

13

Oracle Database Resource Manager



ORACLE®

Objectives for Lesson 13

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

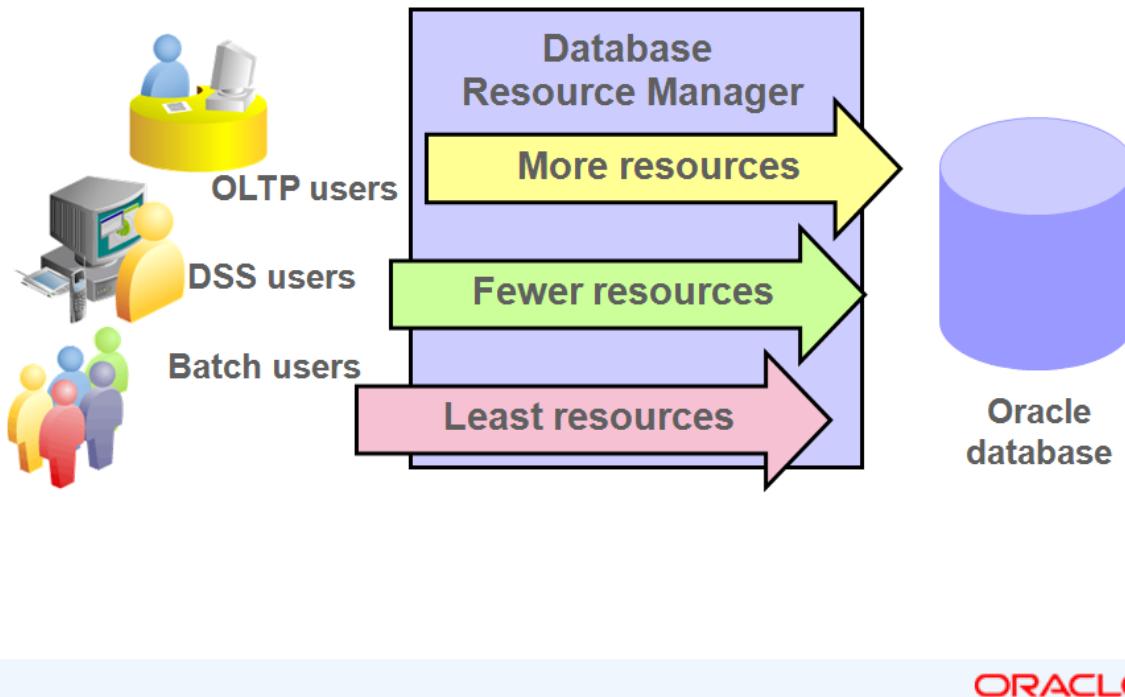
After completing this lesson, you should be able to:

- Configure the Database Resource Manager
- Access and create resource plans
- Create consumer groups
- Specify directives for allocating resources to consumer groups
- Map consumer groups to plans
- Activate a resource plan
- Monitor the Resource Manager



ORACLE®

Oracle Database Resource Manager Overview



ORACLE®

BY COPYING MATERIALS FROM THIS COMPUTER OR IN THIS CLASSROOM. THESE MATERIALS ARE FOR YOUR PERSONAL USE ONLY.

By using the Database Resource Manager (also called the Resource Manager), you have more control over the allocation of machine resources than is normally possible through operating system resource management alone. If resource management decisions are made by the operating system, it can lead to problems such as:

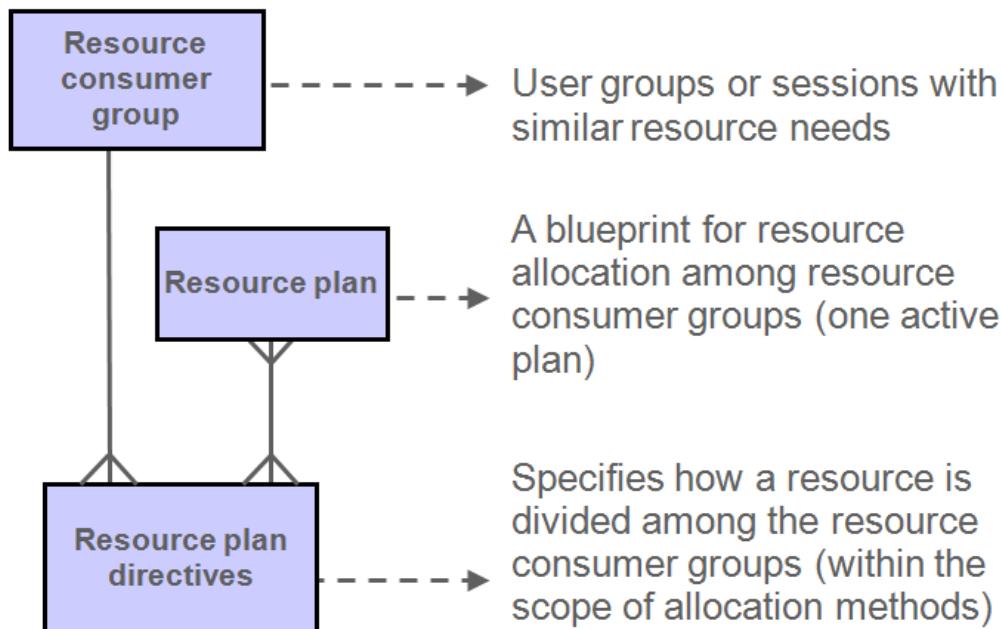
- Excessive overhead resulting from operating system context switching of Oracle Database server processes when the number of server processes is high
- Suspension of a database server process that is holding a latch
- Unequal distribution of resources among all Oracle Database processes, and inability to prioritize one task over another
- Inability to manage database-specific resources, such as parallel execution servers and active sessions

The Database Resource Manager controls the distribution of resources among various sessions by controlling the execution schedule inside the database. By controlling which sessions run and for how long, the Database Resource Manager can ensure that resource distribution matches the plan directive and, therefore, the business objectives. With the Database Resource Manager, you can guarantee groups of users a minimum amount of processing resources regardless of the load on the system and the number of users.

The DBMS_RESOURCE_MANAGER_PRIVS package contains the procedures to grant and revoke the ADMINISTER_RESOURCE_MANAGER system privilege, which is a prerequisite for invoking the Resource Manager.

Resource Manager Elements

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



ORACLE®

Administering systems by using the Resource Manager involves the use of resource plans, resource consumer groups, and resource plan directives.

- A *resource consumer group* defines a set of users or sessions that have similar requirements for using system and database resources.
- A *resource plan* specifies how the resources are distributed among various resource consumer groups. The Resource Manager also allows for creation of plans within plans, called subplans. A PDB resource plan cannot have subplans.
- *Resource plan directives* specify how a particular resource is shared among consumer groups or subplans. You associate resource consumer groups and subplans with a particular resource plan through plan directives.
- *Resource allocation methods* determine what policy to use when allocating for any particular resource. Resource allocation methods are used by resource plans and resource consumer groups.

Using Resource Manager to Allocate Resources

Method	Description
CPU Method	CPU resource allocation among consumer groups and subplans
Degree of Parallelism Limit	Maximum degree of parallelism for any operation within a consumer group
Active Session Pool with Queuing	Limit the number of concurrent active sessions for a consumer group or subplan
Undo Pool	Total undo amount that can be generated by a consumer group or subplan
Execution Time Limit	Maximum execution time for an operation
Idle Time Limit	Amount of time a session can be idle
Consumer Group Switching	Conditions that will cause a consumer group switch
Database Consolidation	Optimize resource allocation among concurrent database sessions
Server Consolidation	Run multiple database instances on the server



The Resource Manager provides several means of allocating resources:

- **CPU Method:** Enables you to specify how CPU resources are allocated among consumer groups and subplans
- **Degree of Parallelism Limit:** Enables you to control the maximum degree of parallelism for any operation within a consumer group
- **Active Session Pool with Queuing:** Allows you to limit the number of concurrent active sessions for a consumer group or subplan. If a group exceeds the maximum allowed number of sessions, new sessions are placed in a queue where they wait for an active session to complete. You can also specify a time limit on how long a session will wait before exiting with an error.
- **Undo Pool:** Enables you to control the total amount of undo that can be generated by a consumer group or subplan. Whenever the total undo space exceeds the amount specified by `UNDO_POOL`, no further `INSERT`, `UPDATE`, or `DELETE` commands are allowed until undo space is freed by another session in the same group or the undo pool is increased for the consumer group. If the consumer group's quota is exceeded during the execution of a DML statement, the operation aborts and returns an error. Queries are still allowed, even if a consumer group has exceeded its undo threshold.
- **Execution Time Limit:** Allows you to specify a maximum execution time allowed for an operation. The Oracle Database server uses cost-based optimizer statistics to estimate how long an operation will take. If it is longer than the maximum time allowed (`MAX_EST_EXEC_TIME`), the operation returns an error and is not started. If a resource consumer group has more than one plan directive with `MAX_EST_EXEC_TIME` specified, the Resource Manager chooses the most restrictive of all incoming values.

- **Idle Time Limit:** Enables you to specify an amount of time for which a session can be idle, after which it will be terminated (`MAX_IDLE_TIME`). You can further restrict the Resource Manager to terminate only those sessions that are blocking other sessions (`MAX_IDLE_TIME_BLOCKER`).
- **Consumer Group Switching:** Specifies conditions that will cause a session to switch consumer groups. Typically, overuse of a resource is specified and a session is switched to a more restrictive consumer group. The session remains in the switched consumer group until it becomes idle, or if directed after the top-level call is completed. Then it will return to the initial consumer group. The initial consumer group is the group that a session is assigned to at login. The top is the current PL/SQL block or each SQL statement that is issued outside a PL/SQL block by the client. You can create a plan directive, so that the Resource Manager automatically switches the user back to the initial consumer group at the end of the top call.
- **Database Consolidation:** The Resource Manager enables you to optimize resource allocation among concurrent database sessions. Database consolidation requires that applications are isolated from each other. If one application experiences an increase in workload, that increase should not affect other applications. In addition, the performance of each application should be consistent. Good candidate applications for database consolidation are automated maintenance tasks because currently, these applications can take up to 100% of the server CPU resources.
- **Server Consolidation:** Because many test, development, and small production databases are unable to fully utilize the servers that they are on, server consolidation provides a possible alternative. With server consolidation, resources are more fully utilized by running multiple database instances on the server. The method for managing CPU allocations on a multi-CPU server with multiple database instances is called Instance Caging. Because Instance Caging is simple to configure and does not require any new software to be licensed or installed, it is an excellent alternative to other server consolidation tools, such as virtualization and O/S workload managers.

You can access resource plans with the graphical interface of Enterprise Manager Cloud Control or the command line of the `DBMS_RESOURCE_MANAGER` package.

The automated maintenance tasks rely on the Resource Manager being enabled during the maintenance windows. When a maintenance window opens, the `DEFAULT_MAINTENANCE_PLAN` resource manager plan is automatically set to control the amount of CPU that is used by the automated maintenance tasks. To be able to give different priorities to each possible task during a maintenance window, various consumer groups are assigned to `DEFAULT_MAINTENANCE_PLAN`.

Creating a Simple Resource Plan

```
BEGIN  
DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN(SIMPLE_PLAN => 'SIMPLE_RESPLAN1',  
CONSUMER_GROUP1 => 'CONSGROUP1', GROUP1_PERCENT => 80,  
CONSUMER_GROUP2 => 'CONSGROUP2', GROUP2_PERCENT => 20);  
END;
```

Consumer Group	Level 1	Level 2	Level 3
SYSGROUP	100%		
CONSGROUP1		80%	
CONSGROUP2		20%	
OTHER_GROUPS			100%



You can create a simple resource plan by using the `DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN` procedure. You can create consumer groups and allocate resources to them by using this single procedure.

The `CREATE_SIMPLE_PLAN` procedure accepts the following arguments:

- `SIMPLE_PLAN`: Name of the plan
- `CONSUMER_GROUP1` (through `CONSUMER_GROUP8`): Consumer group name(s)
- `GROUP1_PERCENT` (through `GROUP8_PERCENT`): CPU resource allocated to the group(s)

The plan uses the default CPU allocation policy (`EMPHASIS`). Each consumer group uses the default scheduling policy (`ROUND_ROBIN`).

Creating a Complex Resource Plan

1. Use `CREATE_PENDING_AREA` to create a pending area.

2. Use `CREATE_CONSUMER_GROUP` to create, modify, or delete consumer groups.

3. Use `SET_CONSUMER_GROUP_MAPPING` to map sessions to consumer groups.

4. Use `CREATE_PLAN` to create the resource plan.

5. Use `CREATE_PLAN_DIRECTIVE` to create resource plan directives.

6. Use `VALIDATE_PENDING_AREA` to validate the pending area.

7. Use `SUBMIT_PENDING_AREA` to submit the pending area.



THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

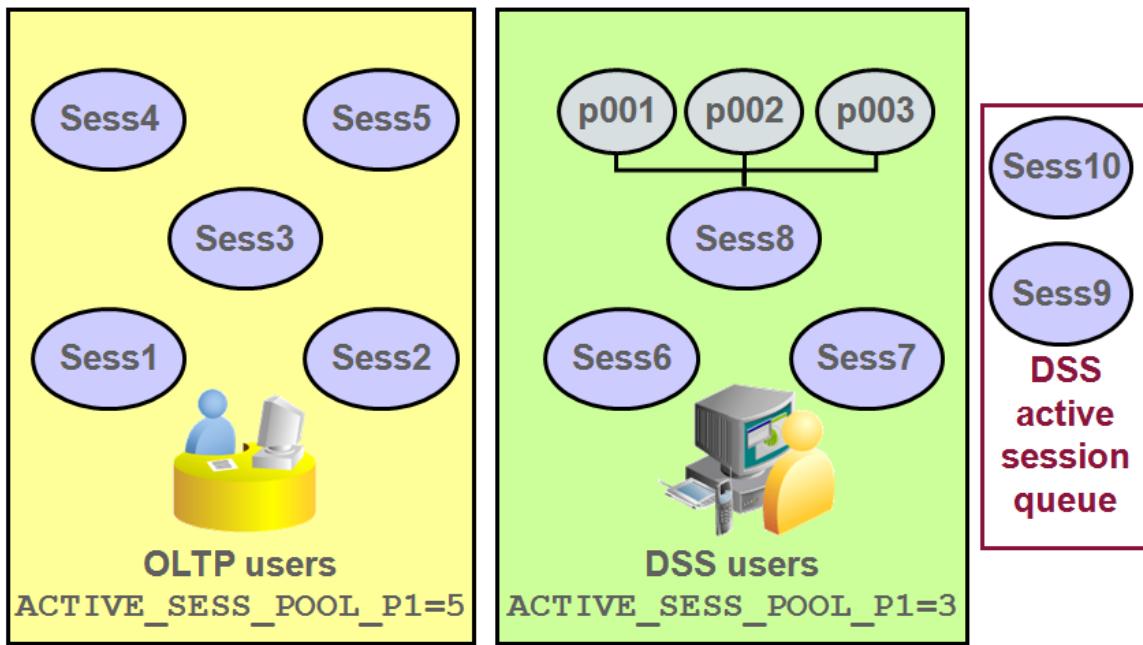
You can perform the following steps by using the named procedures in the `DBMS_RESOURCE_MANAGER` package or by using Enterprise Manager Cloud Control.

1. Create a pending area: To create a new resource plan, or update or delete an existing resource plan, you must create a pending area. The pending area is a staging area where you can create and modify plans without affecting executing applications. After you create the pending area, the Oracle Database server copies existing plans into it so they can be updated if required. Create a pending area by using the `CREATE_PENDING_AREA` procedure.
2. Create, modify, or delete consumer groups: Create a resource consumer group and specify a resource allocation method (ROUND-ROBIN or RUN-TO-COMPLETION) for distributing CPU among the sessions in the consumer group. Use the `CREATE_CONSUMER_GROUP` procedure.
3. Map sessions to consumer groups: Use the `SET_CONSUMER_GROUP_MAPPING` procedure to map a session attribute type and attribute value to a consumer group. The session attribute types include Oracle Database username, database service name, operating system username, client program name, and module name. Refer to the *Oracle Database Administrator's Guide* for a complete list of accepted session attributes. Unassigned sessions are part of the `OTHER_GROUPS` consumer group.
4. Create the resource plan: When you create the resource plan, you provide a name and optionally specify the resource allocation method for CPU (`EMPHASIS` or `RATIO`), active session pool resource allocation method (`ACTIVE_SESS_POOL_ABSOLUTE`), resource allocation method for degree of parallelism (`PARALLEL_DEGREE_LIMIT_ABSOLUTE`), and queuing resource allocation method (`FIFO_TIMEOUT`). Use `CREATE_PLAN` to create a resource plan. Additional information follows in this lesson.

5. Create resource plan directives: Resources are allocated to consumer groups based on the resource plan directives. Use CREATE_PLAN_DIRECTIVE to specify resource plan directives.
6. Create resource plan directives: Resources are allocated to consumer groups based on the resource plan directives. Use CREATE_PLAN_DIRECTIVE to specify resource plan directives.
7. Submit the pending area: After validating the pending area, submit it by using SUBMIT_PENDING_AREA. When you submit the pending area, new and updated plan information is stored in the data dictionary. New plans are not activated when the pending area is submitted. Modified plans are reactivated with their new plan definition.

Using the Active Session Pool Feature

THESE eKIT MATERIALS ARE FOR YOUR PERSONAL USE ONLY. COPYING OR DISTRIBUTION OF THESE eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.



ORACLE®

Using the Active Session Pool feature, you can control the maximum number of concurrently active sessions per resource consumer group. With this functionality, a DBA can indirectly control the amount of resources that any resource consumer group uses because resource consumption is proportional to the number of active sessions. Using an active session pool can help to reduce the number of servers taking resources in the system, thus avoiding inefficient paging, swapping, and other resource depletion (such as memory) resulting from attempting to run too many jobs simultaneously.

After the Active Session Pool is filled with active sessions, the Resource Manager queues all subsequent sessions attempting to become active until other active sessions complete or become inactive. An active session is one currently involved in a transaction, query, or parallel operation. Individual parallel slaves are not counted as sessions; the entire parallel operation counts as one active session.

There is only one queue per resource consumer group and the queuing method is first in, first out (FIFO) with a timeout. The queue is implemented as a memory structure and cannot be queried directly.

LIMITING CPU UTILIZATION AT THE DATABASE LEVEL

Specify minimum and maximum CPU utilization limits.

DB Consolidation Plan #1		
	CPU Allocation	Maximum Utilization Limit
App 1	50%	60%
App 2	20%	30%
App 3	20%	30%
App 4	10%	20%

Specify maximum CPU utilization limits only.

DB Consolidation Plan #2		
	CPU Allocation	Maximum Utilization Limit
App 1	null	50%
App 2	null	20%
App 3	null	20%
App 4	null	10%

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE( -  
    plan          => 'db_consolidation_plan',  
    group_or_subplan => 'App_1',  
    mgmt_p1        => 50,  
    max_utilization_limit => 60) ;
```



For concurrent database sessions: Database consolidation requires that applications are isolated from each other. If one application experiences an increase in workload, that increase should not affect other applications. In addition, the performance of each application should be consistent.

CPU directives can be used to:

- Specify a minimum CPU allocation for each application
- Designate how unused allocations should be redistributed

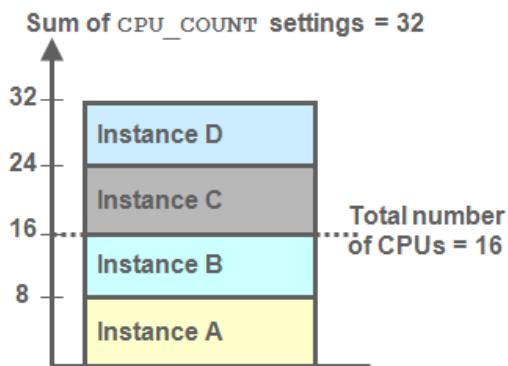
The MAX_UTILIZATION_LIMIT attribute of resource plan directives enables you to impose an absolute upper limit on CPU utilization for a resource consumer group. This absolute limit overrides any redistribution of CPU within a plan.

Good candidate applications for database consolidation are automated maintenance tasks because currently these applications can take up to 100% of the server CPU resources. You can set a maximum limit for each auto-task consumer group.

LIMITING CPU UTILIZATION AT THE SERVER LEVEL

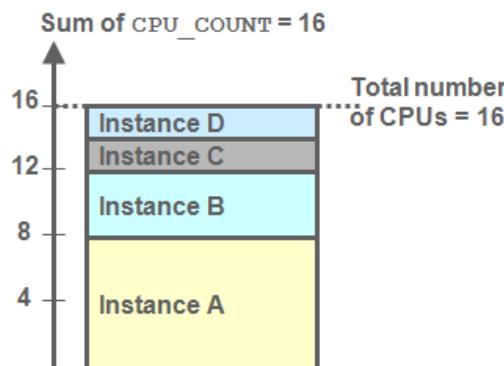
THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM. COPYING OR DISTRIBUTION IS PROHIBITED.

Over-provisioning approach:
One database instance
can still impact the others.



With all four instances active,
one instance can get
 $8 / (8 + 8 + 8 + 8) = 25\%$ of CPU.

Partitioning approach:
One database instance
cannot impact the others.



Each instance has a
dedicated number of CPUs.

ORACLE®

Because many test, development, and small production databases are unable to fully utilize the servers that they are on, *server consolidation* provides a possible alternative. With server consolidation, resources are more fully utilized by running multiple database instances on the server. However, this may bring about CPU contention and an adverse impact due to workload surges on one instance.

Instance caging is a method that uses the `CPU_COUNT` initialization parameter to limit the number of CPUs that an instance can use. In addition, the Resource Manager is employed to allocate the CPUs for the database sessions based on the instance resource plan.

Configure instance caging in two steps, by enabling:

- The Resource Manager, which limits the amount of CPU that the database instance consumes
- The `CPU_COUNT` parameter, which specifies the maximum (the limit), not actual amount of CPU that the database instance can use at any time

By default, the CPU Resource Manager assumes that the database instance can use all CPUs on a server. To enable instance caging, any resource plan with CPU directives can be used.

Over-provisioning approach: This approach is appropriate for noncritical databases and low-load, noncritical production systems. Although the instances impact each other's performance, at any given time, one or more of the instances may be idle or experiencing a low load.

Although the database instances can impact each other's performance, instance caging limits their impact and helps to provide predictable performance. In the example on the left, where all four instances have `CPU_COUNT` set to 8,

the maximum percentage of CPU that a database instance can consume at any point in time is its own limit divided by the sum of the limits for all active databases. In this example, one instance will be able to consume $8 / (8 + 8 + 8 + 8) = 25\%$ of the CPU. If only two instances are active, one instance will be able to consume $8 / (8 + 8) = 50\%$ of the CPU.

Partitioning approach: This approach is appropriate for critical product systems. It prevents the instances from interfering with each other and provides predictable performance.

Instance caging can partition the CPU resources by ensuring that the sum of all CPU limits does not exceed the total number of CPUs. In the example on the right, if four database instances share a 16-CPU server, their limits can be set to 8, 4, 2, and 2. By dedicating CPU resources to a database instance, partitioning provides two advantages:

- One database instance's CPU load cannot affect another.
- Each database instance's CPU resources is fixed, leading to more predictable performance.

Viewing Resource Manager Information

View Name	Information
DBA_RSRC_PLANS	Plans and status
DBA_RSRC_PLAN_DIRECTIVES	Plan directives
DBA_RSRC_CONSUMER_GROUPS	Consumer groups
DBA_RSRC_CONSUMER_GROUP_PRIVS	Users/roles
DBA_RSRC_GROUP_MAPPINGS	Consumer group mapping
DBA_RSRC_MAPPING_PRIORITY	Mapping priority
DBA_USERS	Column INITIAL_RSRC_CONSUMER_GROUP
DBA_RSRC_MANAGER_SYSTEM_PRIVS	Users/roles
DBA_CDB_RSRC_PLANS	All CDB resource plans
DBA_CDB_RSRC_PLAN_DIRECTIVES	CDB resource plan directives

ORACLE®

LIMITING PGA MEMORY ALLOCATED TO EACH SESSION

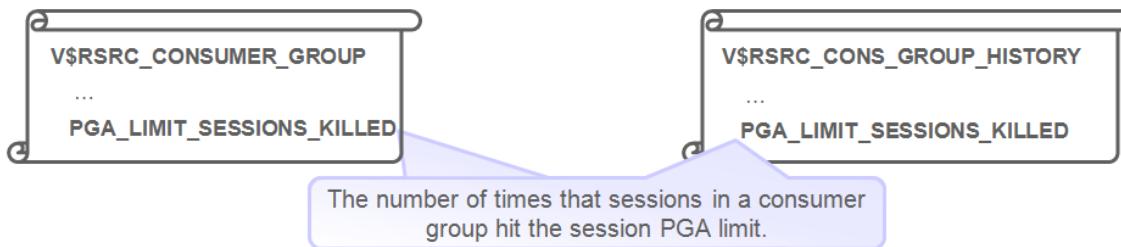
- Avoid excessive usage of PGA memory.
- Set a limit on PGA memory for each session.

DAY plan

Consumer Group	Session_PGA_limit
OLTP	350
REPORTS	100
OTHER	50

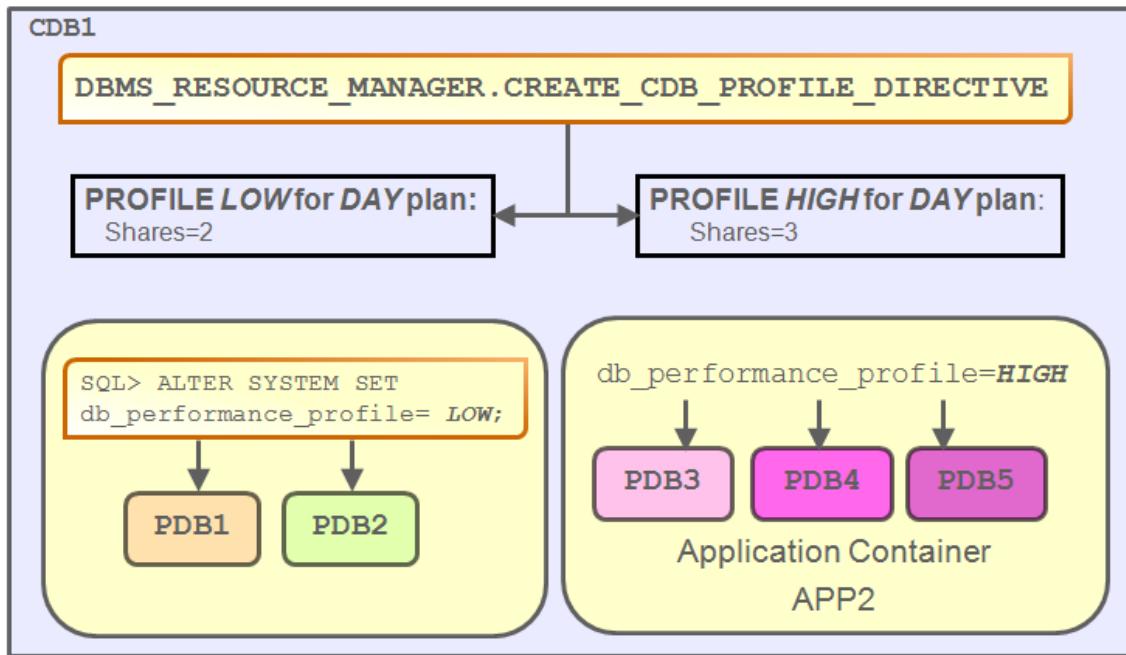
NIGHT plan

Consumer Group	Session_PGA_limit
OLTP	50
REPORTS	400
OTHER	50



ORACLE®

Creating Directives for PDB Performance Profiles



ORACLE®

You can specify Resource Manager directives for a set of PDBs by using PDB performance profiles.

By creating a Resource Manager performance profile, you can set the following resource limits:

- SHARES
- UTILIZATION_LIMIT
- PARALLEL_SERVER_LIMIT

After creating performance profiles for different plans, set the DB_PERFORMANCE_PROFILE initialization parameter for each PDB to the appropriate Resource Manager performance profile.

Summary for Lesson 13

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

In this lesson, you should have learned to:

- Configure the Database Resource Manager
- Access and create resource plans
- Create consumer groups
- Specify directives for allocating resources to consumer groups
- Map consumer groups to plans
- Activate a resource plan
- Monitor the Resource Manager



ORACLE®

Practice 13 Overview

- 13-1: Dispatching Resources Amongst PDBs by Using Resource Manager
- 13-2: Avoiding Excessive PGA Memory Usage



Practice 13-1 Dispatching Resources Amongst PDBs by Using Resource Manager

Overview

In this practice, you dispatch the ratio of CPU usage amongst concurrent database sessions in the two PDBs, PDB1 and PDB2. You create two CDB Resource Manager plans and associated directives to limit CPU resources used by the two PDBs.

Tip:

In a CDB with multiple PDBs, some PDBs typically are more important than others. The Resource Manager enables you to prioritize the resource usage of specific PDBs; for example, CPU, I/O, and allocation of parallel execution slaves in the context of parallel statement queuing. Resource Manager grants different PDBs different shares of the system resources so that more resources are allocated to the more important PDBs. In addition, limits can be used to restrain the system resource usage of specific PDBs.

Assumptions

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

Tasks

Create CPU Burner Procedures in PDB1 and PDB2

In this section, you create a PL/SQL procedure in both PDB1 and PDB2 that burns CPU resources.

1. Open a new terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window-ROOT.

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

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

2. Execute the `$HOME/labs/TUNING_rsrc_cleanup.sh` shell script to create PDB1 and PDB2.

```
$ $HOME/labs/TUNING_rsrc_cleanup.sh

...
$
```

3. In PDB1, create the PL/SQL procedure as the `SYSTEM` user.

- a. Open another terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window-PDB1.

```
$ . 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 SYSTEM user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus SYSTEM/<password>@PDB1  
  
...  
SQL>
```

- c. Execute the `create_burn_cpu.sql` script.

```
SQL> @$HOME/labs/create_burn_cpu.sql  
  
Procedure created.  
SQL>
```

4. In PDB2, create the same PL/SQL procedure as the SYSTEM user.

- a. Open another terminal window and source the `oraenv` script. For the `ORACLE_SID` value, enter `ORCL`. This window will be referred to as window-PDB2.

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

- b. Start SQL*Plus and connect to PDB2 as the SYSTEM user. See Appendix - Product-Specific Credentials for the password.

```
$ sqlplus SYSTEM/<password>@PDB2  
  
...  
SQL>
```

- c. Execute the `create_burn_cpu.sql` script.

```
SQL> @$HOME/labs/create_burn_cpu.sql  
  
Procedure created.  
SQL>
```

Create Resource Manager Plans in the CDB

In this section, you return to window-ROOT and create two Resource Manager plans in the CDB called FAIRPLAN and UNFAIRPLAN.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

FAIRPLAN should give one share to both PDB1 and PDB2, and UNFAIRPLAN should give one share to PDB1 and five shares to PDB2. You then activate FAIRPLAN and set SERVEROUTPUT equal to ON for both PDB1 and PDB2.

1. In window-ROOT, review the DBMS_RESOURCE_MANAGER procedures that are used to create the plans. They are as follows:

```
$ cat $HOME/labs/setup_plan_directives.sql

EXEC DBMS_Resource_Manager.Clear_Pending_Area()
EXEC DBMS_Resource_Manager.Create_Pending_Area()
EXEC DBMS_Resource_Manager.Create_CDB_Plan('fairplan', 'One share each')
EXEC DBMS_Resource_Manager.Create_CDB_Plan_Directive('fairplan', 'pdb1', shares => 1)
EXEC DBMS_Resource_Manager.Create_CDB_Plan_Directive('fairplan', 'pdb2', shares => 1)
EXEC DBMS_Resource_Manager.Create_CDB_Plan('unfairplan', 'one share to pdb1 and five to pdb2')
EXEC DBMS_Resource_Manager.Create_CDB_Plan_Directive('unfairplan', 'pdb1', shares => 1)
EXEC DBMS_Resource_Manager.Create_CDB_Plan_Directive('unfairplan', 'pdb2', shares => 5)
EXEC DBMS_Resource_Manager.Validate_Pending_Area()
EXEC DBMS_Resource_Manager.Submit_Pending_Area()
```

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

```
$ sqlplus / AS SYSDBA

...
SQL>
```

3. Execute the \$HOME/labs/setup_plan_directives.sql to create the plans in the CDB root.

```
SQL> @$HOME/labs/setup_plan_directives.sql

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

4. Ensure that both plans and associated directives were created correctly.

- a. List the Resource Manager plans in the CDB root. The result shows FAIRPLAN and UNFAIRPLAN.
Note: The ID for the CDB root is 1.

```
SQL> SELECT plan FROM cdb_cdb_rsrc_plans
  2 WHERE con_id=1 AND plan IN ('FAIRPLAN', 'UNFAIRPLAN');

PLAN
-----
FAIRPLAN
UNFAIRPLAN
SQL>
```

- b. List the shares of system resources granted to PDB1 and PDB2.

Note: If you prefer, you can enter @\${HOME}/labs/RM_1.sql to run a SQL script instead of typing the following code. The results below are formatted for easier viewing.

```
SQL> SELECT plan, pluggable_database, shares
  2 FROM   cdb_cdb_rsrc_plan_directives
  3 WHERE  con_id = 1 AND plan in ('FAIRPLAN', 'UNFAIRPLAN') AND
pluggable_database in ('PDB1', 'PDB2')
  4 ORDER BY 1, 2;

PLAN          PLUGGABLE_DATABASE      SHARES
-----
FAIRPLAN        PDB1                  1
FAIRPLAN        PDB2                  1
UNFAIRPLAN     PDB1                  1
UNFAIRPLAN     PDB2                  5
SQL>
```

5. Activate FAIRPLAN in the CDB root.

```
SQL> ALTER SYSTEM SET resource_manager_plan = fairplan;
System altered.

SQL>
```

6. Display the names of all currently active resource plans in the CDB root by querying the V\$RSRC_PLAN view. The result shows FAIRPLAN.

```
SQL> SELECT name FROM v$rsrc_plan WHERE con_id = 1;

NAME
-----
FAIRPLAN
SQL>
```

Execute the CPU Burner Procedures

In this section, you have races between PDB1 and PDB2. Be prepared to work quickly from step to step and between terminal windows. The first race you have is executing the CPU burner procedure (which you created earlier) on both

PDB1 and PDB2 with FAIRPLAN activated in the CDB. The second race you have is executing the same procedure on both PDBs, but with UNFAIRPLAN activated. Find out whether the different resource shares in the plans make a difference in the outcomes.

1. Enable server output in the PDBs.

- In window-PDB1, set SERVEROUTPUT equal to ON for PDB1.

```
SQL> SET SERVEROUTPUT ON
```

```
SQL>
```

- In window-PDB2, set SERVEROUTPUT equal to ON for PDB2.

```
SQL> SET SERVEROUTPUT ON
```

```
SQL>
```

2. Execute the following two steps back to back as quickly as possible to begin the first race. The procedure takes about a minute to run.

- In window-PDB1, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
```

```
CPU:    74.4 Wall: 149.0 k: 2000000000  
PL/SQL procedure successfully completed.  
SQL>
```

- In window-PDB2, execute the CPU burner procedure.

```
SQL> EXEC Burn_CPU_For_RM_Demo()
```

```
CPU:    74.4 Wall: 144.2 k: 2000000000  
PL/SQL procedure successfully completed.  
SQL>
```

3. Question: What do you observe?

Answer: Both procedures finish their execution almost at the same time, and have both consumed almost the same CPU and wall-clock time during their execution. This outcome is expected because with FAIRPLAN activated, each PDB is receiving one share of CPU.

4. In window-ROOT, change the Resource Manager plan to UNFAIRPLAN.

```
SQL> ALTER SYSTEM SET resource_manager_plan = unfairplan;
```

```
System altered.  
SQL>
```

5. Display the names of all currently active resource plans in the CDB root by querying the V\$RSRC_PLAN view. The result shows UNFAIRPLAN.

```
SQL> SELECT name FROM v$rsrc_plan WHERE con_id = 1;  
  
NAME  
-----  
UNFAIRPLAN  
SQL>
```

6. Enable server output in the PDBs again to prepare for the second race.

- In window-PDB1, set SERVEROUTPUT equal to ON for PDB1.

```
SQL> SET SERVEROUTPUT ON  
  
SQL>
```

- In window-PDB2, set SERVEROUTPUT equal to ON for PDB2.

```
SQL> SET SERVEROUTPUT ON  
  
SQL>
```

7. Execute the following two steps back to back as quickly as possible to begin the second race.

- In window-PDB1, execute the CPU burner procedure.

```
SQL> execute Burn_CPU_For_RM_Demo()  
  
CPU:    74.0 Wall: 148.9 k: 2000000000  
PL/SQL procedure successfully completed.  
SQL>
```

- In window-PDB2, execute the CPU burner procedure.

```
SQL> execute Burn_CPU_For_RM_Demo()  
  
CPU:    74.1 Wall: 89.1 k: 2000000000  
PL/SQL procedure successfully completed.  
SQL>
```

- Question: What do you observe?

Answer: Now, execution of the CPU burner procedure takes much longer to execute in PDB1 than it does in PDB2. This is expected because PDB2 is assigned five shares whereas PDB1 is assigned only one. However, the difference is not five times slower because after the procedure is executed in PDB2, all CPU cycles go to PDB1.

Reset the Resource Manager Plan Back to Its Default

- In window-ROOT, change the Resource Manager plan back to its default.

```
SQL> ALTER SYSTEM SET resource_manager_plan = '';  
System altered.  
SQL>
```

- Display the names of all currently active resource plans in the CDB root by querying the V\$RSRC_PLAN view. The result shows ORA\$INTERNAL_CDB_PLAN.

```
SQL> SELECT name FROM v$rsrc_plan WHERE con_id = 1;  
NAME  
-----  
ORA$INTERNAL_CDB_PLAN  
SQL>
```

- Exit SQL*Plus in window-ROOT, window-PDB1, and window-PDB2, and close all the terminal windows.

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

Practice 13-2 Avoiding Excessive PGA Memory Usage

Overview

In this practice, you use Resource Manager to avoid excessive PGA memory consumption in PDB1.

Assumptions

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

Tasks

Start a Reporting Query in Window 1

In this section, you connect to PDB1 as the `OE` user and run a query that uses too much PGA memory.

1. Open a new terminal window and execute the `$HOME/labs/UNDO_setup_tuning.sh` shell script to create the appropriate tables in PDB1. You can ignore errors when you run the script because they are expected. This window will be referred to as window 1.

```
$ $HOME/labs/UNDO_setup_tuning.sh  
...  
$
```

2. 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 `OE` user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus oe/<password>@PDB1  
...  
SQL>
```

4. Execute a reporting query on the `OE.LINEORDER` table. The query will run continuously while you work through the next steps.

```
SQL> SELECT * FROM oe.lineorder  
  2 ORDER BY lo_orderkey, lo_custkey, lo_partkey, lo_suppkey;  
...  
SQL>
```

Check and Limit the PGA Memory Usage in Window 2

Some users informed you that their query performance decreased in PDB1 while OE was generating some reporting queries on the LINEORDER table. In this section, you check the current PGA memory usage for the OE user in PDB1, and then use the Resource Manager directive to limit the PGA memory usage.

1. Open another terminal window and source the oraenv script. For the ORACLE_SID value, enter ORCL. This window will be referred to as window 2.

```
$ . 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 SYSTEM user. See [Appendix - Product-Specific Credentials](#) for the password.

```
$ sqlplus SYSTEM/<password>@PDB1

...
SQL>
```

3. Execute the following query to check the current PGA memory used by the OE user. The results show that the SESSION PGA MEMORY MAX value, which is the peak PGA size for the session, is about 110MB (your values may be different than those shown below). You decide to reduce this number to 80MB. The SESSION PGA MEMORY value is the current PGA size for the session.

Note: If you prefer, you can enter @\$HOME/labs/PGAMEM_1.sql to run a SQL script instead of typing the following code.

```
SQL> COLUMN value FORMAT 9999999999999999
SQL> SELECT s.username, a.name, m.value
  2  FROM v$sesstat m, v$statname a, v$session s
  3  WHERE m.statistic# = a.statistic# AND m.sid = s.sid
  4  AND a.name LIKE 'session pga memory%'
  5  AND username = 'OE';

USERNAME          NAME          VALUE
-----          -----
OE                session pga memory      4360376
OE                session pga memory max  110897624
SQL>
```

4. Execute the SQL script \$HOME/labs/script_rm.sql to create a Resource Manager plan called pga_plan, create a consumer group named reporting_users, and associate the OE user to the reporting_users group. The results below are formatted for easier viewing.

```
SQL> @$HOME/labs/script_rm.sql  
...  
PLAN      GROUP_OR_SUBPLAN      SESSION_PGA_LIMIT  
-----  
-----  
PGA_PLAN  OTHER_GROUPS        200  
SQL>
```

- Set the PGA limit for the reporting_users consumer group to 80MB by executing the following PL/SQL procedures.

```
SQL> EXEC dbms_resource_manager.clear_pending_area()  
PL/SQL procedure successfully completed.  
  
SQL> EXEC dbms_resource_manager.create_pending_area()  
PL/SQL procedure successfully completed.  
  
SQL> EXEC dbms_resource_manager.create_plan_directive(  
      'PGA_plan','Reporting_Users',session_pga_limit => 80)  
PL/SQL procedure successfully completed.  
  
SQL> EXEC dbms_resource_manager.validate_pending_area()  
PL/SQL procedure successfully completed.  
  
SQL> EXEC dbms_resource_manager.submit_pending_area()  
PL/SQL procedure successfully completed.  
SQL>
```

- Activate the plan in PDB1.

```
SQL> ALTER SYSTEM SET resource_manager_plan='PGA_plan';  
System altered.  
SQL>
```

- Flush the buffer cache.

```
SQL> ALTER SYSTEM FLUSH buffer_cache;  
System altered.  
SQL>
```

- Verify that pga_plan is set. The results are formatted for easier viewing.

```
SQL> SELECT plan, group_or_subplan, session_pga_limit
  2  FROM dba_rsrc_plan_directives
  3  WHERE plan = 'PGA_PLAN';

PLAN      GROUP_OR_SUBPLAN      SESSION_PGA_LIMIT
-----      -----
PGA_PLAN    OTHER_GROUPS          200
PGA_PLAN    REPORTING_USERS       80

SQL>
```

Re-execute the Query and Check the PGA Memory Usage

1. In window 1, stop the query by pressing **Ctrl+c**.
2. In window 1, start a new session.
 - a. Exit SQL*Plus.

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

- b. Start SQL*Plus and connect to PDB1 as the OE user. See [Appendix D - Product-Specific Credentials](#) for the password.

```
$ sqlplus oe/<password>@PDB1
...
SQL>
```

3. In window 1, rerun the query on the OE.LINEORDER table. Recall that in window 1, you are connected to PDB1 as the OE user. You will get an error message stating that the PGA limit (80MB) has exceeded and that the process is terminated.

```
SQL> SELECT * FROM oe.lineorder
      ORDER BY lo_orderkey, lo_custkey, lo_partkey, lo_suppkey;

SELECT * FROM oe.lineorder
*
ERROR at line 1:
ORA-10260: PGA limit (80 MB) exceeded - process terminated
SQL>
```

4. In window 2, query the number of sessions killed because of exceeding the PGA limit. The result shows that one session was killed for the REPORTING_USERS resource consumer group. This explains the error you received in the previous step. The order of your results may be different than the order shown below.

```
SQL> SELECT name, pga_limit_sessions_killed FROM v$rsrc_consumer_group;

NAME          PGA_LIMIT_SESSIONS_KILLED
-----
REPORTING_USERS          1
OTHER_GROUPS            0
SQL>
```

5. In window 2, check the PGA memory limit for the OE user. Recall that in window 2, you are connected to PDB1 as the SYSTEM user. Notice that the SESSION PGA MEMORY MAX value is now around 80 MB. Your values may be slightly different than those shown below.

Note: If you prefer, you can enter @\${HOME}/labs/PGAMEM_1.sql to run a SQL script instead of typing the following code.

```
SQL> COLUMN value FORMAT 9999999999999999
SQL> SELECT s.username, a.name, m.value
  2  FROM v$sesstat m, v$statname a, v$session s
  3 WHERE m.statistic# = a.statistic# AND m.sid = s.sid
  4 AND a.name LIKE 'session pga memory%'
  5 AND username = 'OE';

USERNAME      NAME          VALUE
-----        -----
OE           session pga memory    1935544
OE           session pga memory max 83855544
SQL>
```

6. In windows 1 and 2, exit SQL*Plus.

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

7. In window 2, execute the following script to clean up the Resource Manager plan. Close the terminal window when the script is completed.

```
$ ${HOME}/labs/cleanup_rm.sh

System altered.
PL/SQL procedure successfully completed.
$
```

8. Close both terminal windows.

14

Enterprise Manager Cloud Control



ORACLE®

Objectives for Lesson 14

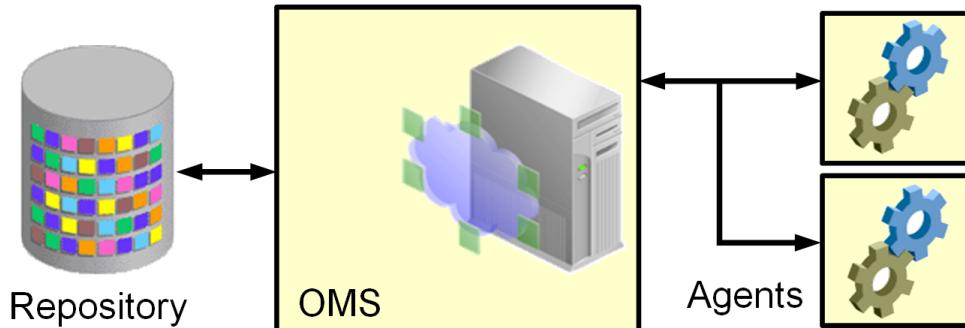
After completing this lesson, you should be able to:

- Identify the utilities that control the Enterprise Manager Cloud Control framework
- Start and stop the Enterprise Manager Cloud Control framework



ORACLE®

Controlling the Enterprise Manager Cloud Control Framework



Component Control Utilities		
Repository	OMS	Agent
SQL*Plus or Server Control	Enterprise Manager Control	Enterprise Manager Control
Listener Control		

ORACLE®

The main components of the Enterprise Manager Cloud Control framework are the following:

- Oracle Management Repository (OMR)
- Oracle Management Server (OMS)
- Oracle Management Agent (OMA)

Each component of the Enterprise Manager Cloud Control framework has its own utility or utilities, as illustrated above, that can be used to monitor, start, and stop the component. In many cases, these utilities also provide some capability to configure the component beyond the simple start-and-stop functionality.

Real Application Cluster (RAC) databases require the use of Server Control commands. For single instances, there is a choice between SQL*Plus and Server Control. Server Control can be used when Oracle Restart is installed and the database is registered with the Oracle Local Registry (OLR).

To start and stop the listener, use commands in the Server Control utility (as shown below) or the Listener Control utility.

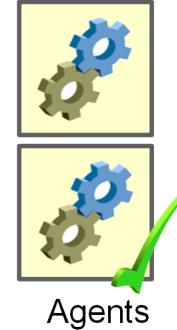
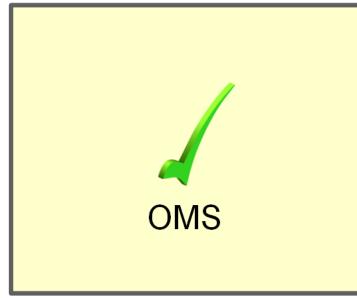
```
srvctl stop database -d orcl -o immediate
srvctl start database -d orcl -o open
```

Starting the Enterprise Manager Cloud Control Framework

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

To start the Cloud Control framework:

1. Start the repository database listener.
2. Start the repository database instance.
3. Start Oracle Management Server (OMS).
4. Start the agent on the OMS server.
5. Start the agents on the managed servers.



ORACLE®

To start the whole Enterprise Manager Cloud Control framework, perform the following steps:

1. Start the repository listener:

```
$ORACLE_HOME/bin/lsnrctl start
```

2. Start the repository database instance:

```
$ORACLE_HOME/bin/sqlplus / as sysdba  
SQL> STARTUP
```

3. Start Oracle Management Server (OMS):

```
$OMS_HOME/bin/emctl start oms
```

4. Start the agent on the OMS server:

```
$AGENT_HOME/bin/emctl start agent
```

5. Start the agent on the managed servers:

```
$AGENT_HOME/bin/emctl start agent
```

Note: Use the SRVCTL command if you have a RAC instance for the repository or the repository is controlled by Oracle Restart.

Stopping the Enterprise Manager Cloud Control Framework

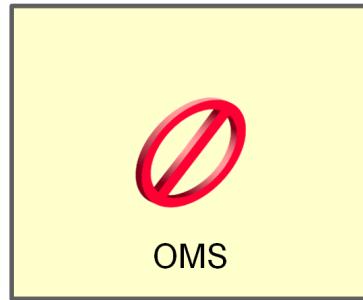
THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

To stop the Enterprise Manager Cloud Control framework:

1. Stop the agents on the managed servers.
2. Stop the agent on the Oracle Management Server (OMS) server.
3. Stop OMS.
4. Stop the repository database instance.



Agents



OMS



Repository

ORACLE®

To stop the whole Enterprise Manager Cloud Control framework, perform the following steps:

1. Stop the agent on the managed servers:

```
$AGENT_HOME/bin/emctl stop agent
```

2. Stop the agent on the Oracle Management Server (OMS) host:

```
$AGENT_HOME/bin/emctl stop agent
```

3. Stop OMS:

```
$OMS_HOME/bin/emctl stop oms
```

4. Stop the repository database instance:

```
$ORACLE_HOME/bin/sqlplus / as sysdba  
SQL> SHUTDOWN IMMEDIATE
```

Note: Use the SRVCTL command if you have a RAC instance for the repository.

Summary for Lesson 14

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED.

In this lesson, you should have learned to:

- Identify the utilities that control the Enterprise Manager Cloud Control framework
- Start and stop the Enterprise Manager Cloud Control framework



ORACLE®

Practice 14 Overview

- 14-1: Registering a Database with EM Cloud Control
- 14-2: Using ADDM

ORACLE®

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING OR TRANSFERRING MATERIALS FROM THIS COMPUTER IS PROHIBITED.

Practice 14-1 Registering a Database with Enterprise Manager Cloud Control

Overview

In this practice, you use Oracle Enterprise Manager Cloud Control (EM Cloud Control) to register an Oracle database. Cloud Control is installed on VM2 and the Oracle database, named `ORCL`, is located on VM1.

Tip:

Tips for navigating EM Cloud Control:

- Use drop-down menus to navigate from one place to another in the product.
- Choose your own home page: When you first log in to EM Cloud Control, you are provided with a selection of pre-defined home pages based on roles. If you are managing databases, you can choose the database home page. If those are not suitable, you can select any page to be your home page instead.
- Mark any page as a “favorite” for quick access. For example, if there are certain targets that you manage quite often, you can mark the page you manage them from as a favorite in the same way you mark a favorite in a browser. However, because the favorites you mark in EM Cloud Control are stored in the repository, you can move from client machine to client machine and your favorites are still available to you.

Assumptions

You have access to two VMs:

- VM1: Has Oracle Database 12.2 installed with one CDB and one PDB (`PDB1`)
- VM2: Has Oracle Database 12.1 installed and EM Cloud Control 13c

Tasks

Connect to EM Cloud Control in VM2

1. Open two TigerVNC sessions:
 - One that connects to VM1, where the 12.2 database resides
 - One that connects to VM2, where EM Cloud Control resides

Tip: To open more than one VNC session at a time, right-click the top border of an existing VNC window and select **New Connection** to initiate a new connection.
2. On VM2, log in to EM Cloud Control.
 - a. Open Firefox.
 - b. If the EM Cloud Control web page is not displayed automatically, enter its URL: `https://localhost:7802/em`.
Note: The format for this URL is `https://<machine_name>:<port_number>/em`.
 - c. If you get an “Untrusted Connection” message (or something similar depending on the browser and version), add an exception and accept the certificate.
 - d. On the EM Cloud Control login page, enter the user name `SYSMAN`, enter the password, and click **Login**. See [Appendix - Product-Specific Credentials](#) for the password.

3. On VM2, review the Enterprise Summary page.

The screenshot shows the Oracle Enterprise Manager Cloud Control 13c interface. The main header reads "ORACLE Enterprise Manager Cloud Control 13c". The top navigation bar includes icons for refresh, search, and notifications, along with "SYSMAN". The title "Enterprise Summary" is at the top left. A pie chart under "Status" indicates the distribution of target status: Up (17), Down (6), and Unknown (12). Below the chart, there's a section for "Incidents" showing one incident updated in the last 24 hours and one in the last 7 days. To the right, there are two expandable sections: "Inventory and Usage" (under "Systems Infrastructure PDU") which displays a message "No data to display", and "Compliance Summary" (under "Targets") which shows a table with columns "Name" and "Average Compliance Score (%)".

[Install the Cloud Control Agent on VM1 from VM2](#)

Continue to work on VM2.

1. In the Setup drop-down list (the cog wheel on the toolbar), select **Add Target**, and then **Add Targets Manually**.
2. If needed, expand the **Overview** section.
3. Under Add Host Targets, click **Install Agent on Host**. The next page may take a moment to appear.

The screenshot shows the "Add Targets Manually" wizard. The title "Add Targets Manually" is at the top. Below it, a section titled "Overview" is expanded. Within this section, there is a sub-section titled "Add Host Targets" featuring a blue server icon. At the bottom of this sub-section, there are two buttons: "Install Agent on Host" (which is highlighted with a red border) and "Install Agent Results".

4. On the "Add Host Targets - Host and Platform" page, do the following, and click **Next**.
- Retain the default session name.
 - Under Agent Software Options, in the Add drop-down list, select **Manually**.
 - In the Host box, enter **12cr2db.example.com**, which is the host name of VM1.
 - In the Platform drop-down list, select **Linux x86-64**.

The screenshot shows the 'Add Target' wizard in Oracle Enterprise Manager Cloud Control 13c. The current step is 'Step 1 of 3'. The session name is 'ADD_HOST_SYSMAN_Aug_15_2016_6:55:37_PM_GMT'. The 'Agent Software Options' section is expanded, showing an 'Add' button and a 'Remove' button. A table lists one host entry:

Host	Platform
12cr2db.example.com	Linux x86-64

5. On the "Add Host Targets - Installation Details" page, do the following, and click **Next**.
- Click the **Collapse Main Section** button (small arrow button on the right, below the table) to view the rest of the Linux x86-64: Agent Installation Details section.
 - In the Installation Base Directory box, enter `/u01/app/oracle/product/agent`. The installer will create this directory on VM1.
 - Click the Instance Directory box. The path `/u01/app/oracle/product/agent/agent_inst` is entered automatically.
 - Next to the Named Credential box, click the **Add a new Named Credential** button (plus sign).
 - In the "Create new Named Credential" dialog box, enter **oracle** in the UserName box. In the Password and Confirm Password boxes, enter the operating system password for the **oracle** user. In the Run

Privilege drop-down list, select **Sudo**. In the "Run as" box, enter **root**. Retain the default value in the Save As box; for example, **NC_HOST_2016-08-15-190109**. Click **OK**.

Create new Named Credential

Enter the user name and password you want to save as a Named Credential.

* **UserName**: oracle

* **Password**: *****

* **Confirm Password**: *****

Run Privilege: Sudo **Run as**: root

Save As: NC_HOST_2016-08-15-190109

OK **Cancel**

- f. In the Privileged Delegation Setting box, retain the default setting `/usr/bin/sudo -u %RUNAS% %COMMAND%`.
- g. In the Port box, retain the default port number **3872**.

Add Target

Host and Platform Installation Details Review

Add Host Targets: Installation Details

On this screen, select each row from the following table and provide the installation details in the Installation Details section.

Linux x86-64: Agent Installation Details

* Installation Base Directory	/u01/app/oracle/product/agent
* Instance Directory	/u01/app/oracle/product/agent/agent_inst
* Named Credential	NC_HOST_2016-08-15-190109(SYSMAN) <input type="button" value="+"/>
Privileged Delegation Setting	/usr/bin/sudo -u %RUNAS% %COMMAND%
Port	3872

6. On the Review page, scroll down to view the configuration details, and click **Deploy Agent**.

ORACLE® Enterprise Manager Cloud Control 13c

SYSMAN ▾ ...

Add Target

Host and Platform Installation Details Review

Add Host Targets: Review

Back Step 3 of 3 Next Deploy Agent Cancel

Review the details you have provided for this deployment session and click Deploy Agent.

Session Name ADD_HOST_SYSMAN_Aug_15_2016_6:55:37_PM_GMT

Deployment Type Fresh Agent Install

OMS Host em13c.example.com

OMS Upload Port 4903

Host Information

Linux x86-64

Hosts 12cr2db.example.com

Agent Software Version 13.1.0.0.0

- View the messages as the deployment is in progress. It takes a few minutes to deploy.

ORACLE® Enterprise Manager Cloud Control 13c

SYSMAN ▾ ...

Add Host

Page Refreshed Aug 15, 2016 7:12:39 PM GMT

Agent Deployment in Progress Refresh Frequency 30 seconds Refresh Cancel

Agent Deployment Summary: ADD_HOST_SYSMAN_Aug_15_2016_6:55:37_PM_GMT

Platform	Host	Initialization	Remote Prerequisite Check	Agent Deployment
Linux x86-64	12cr2db.example.com	✓	✓	✗

Agent Deployment Details: 12cr2db.example.com

Initialization Phase Name	Status	Error	Cause	Recommendation
Remote Validations	✓			
Transferring Agent Software to Destination Host	✓			

- When the message "Agent Deployment Succeeded" is displayed at the top of the page, click **Done**.

✓ Agent Deployment Succeeded

Done

Agent Deployment Summary: ADD_HOST_SYSMAN_Aug_15_2016_6:55:37_PM_GMT

Platform	Host	Initialization	Remote Prerequisite Check	Agent Deployment
Linux x86-64	12cr2db.example.com	✓	✓	✓

Discover the ORCL CDB on VM1 as a Monitored Target in EM Cloud Control

Continue to work on VM2 in EM Cloud Control.

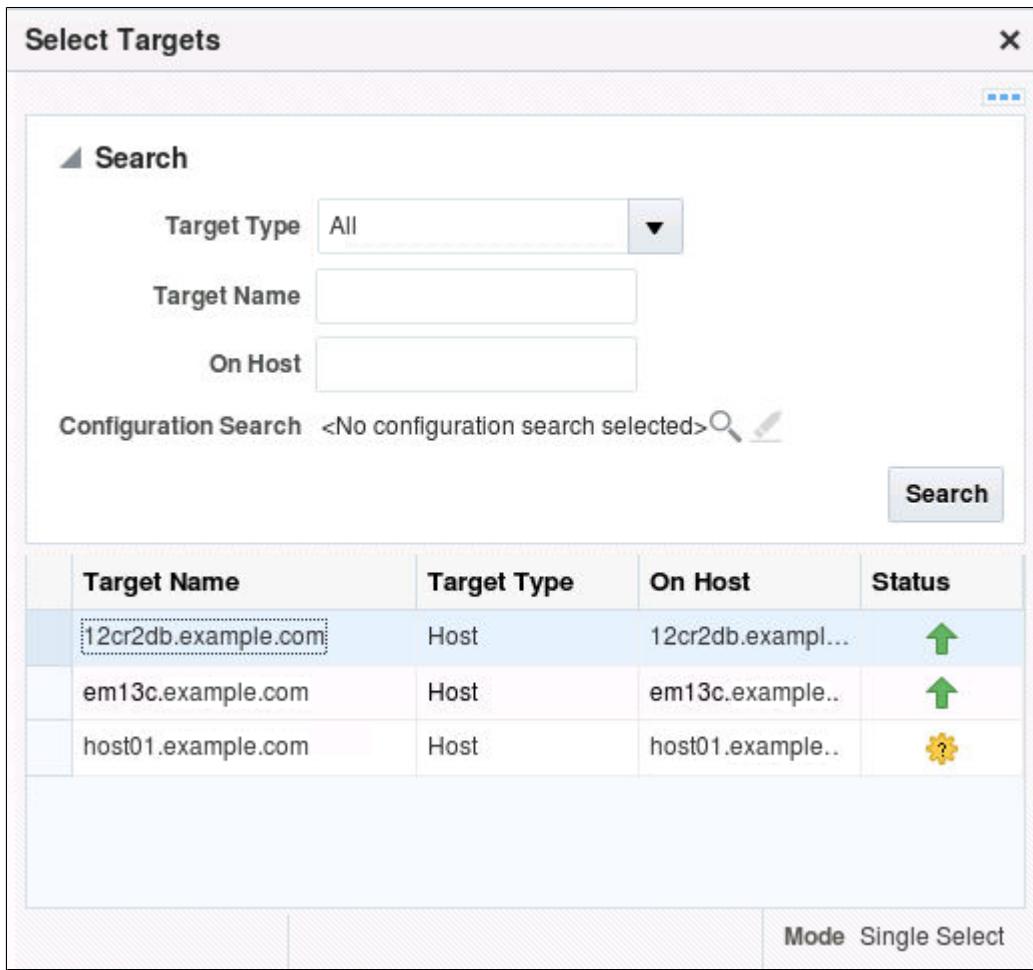
1. In the Targets drop-down list (the bullseye on the toolbar), select **Databases**.
2. Select the **Search List** option.
3. In the Add drop-down list, select **Oracle Database**.

The screenshot shows the 'Databases' search interface. At the top, there are navigation links: Performance, Availability, Security, Schema, and Administration. Below them are two radio buttons: 'View' (unchecked) and 'Database Load Map' (unchecked). The 'Search' section contains a 'Find' dropdown set to 'Name' and a search input field with a magnifying glass icon. Below the search bar are buttons for 'View', 'Add' (highlighted with a red box), 'Remove', and 'Configure'. A table below lists databases by Name, with one entry labeled 'Oracle Database' also highlighted with a red box.

4. To the right of the "Specify Host or Cluster" box, click the **Specify Host or Cluster** button (magnifying glass).

The screenshot shows the 'Database Discovery: Search Criteria' step. It has three tabs: 'Search Criteria' (selected), 'Results', and 'Review'. The main area displays the instruction: 'In order to add targets to be monitored by Enterprise Manager, you must first specify the host or cluster on which'. Below this is a form field with a required indicator (*), the label 'Specify Host or Cluster', and a magnifying glass icon to its right, which is highlighted with a red box.

5. In the Select Targets dialog box, select the target named **12cr2db.example.com**, and click **Select**.

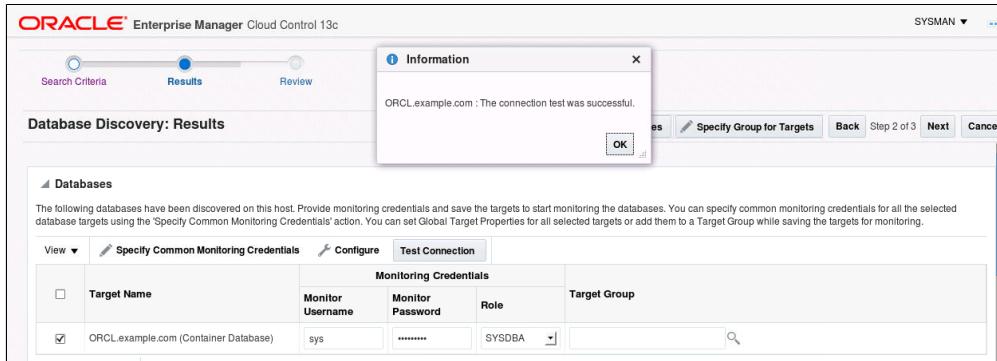


6. In the "Specify Host or Cluster" box, ensure that your VM1 host name is displayed, and click **Next**.



7. On the "Database Discovery: Results" page, do the following, and click **Next**.

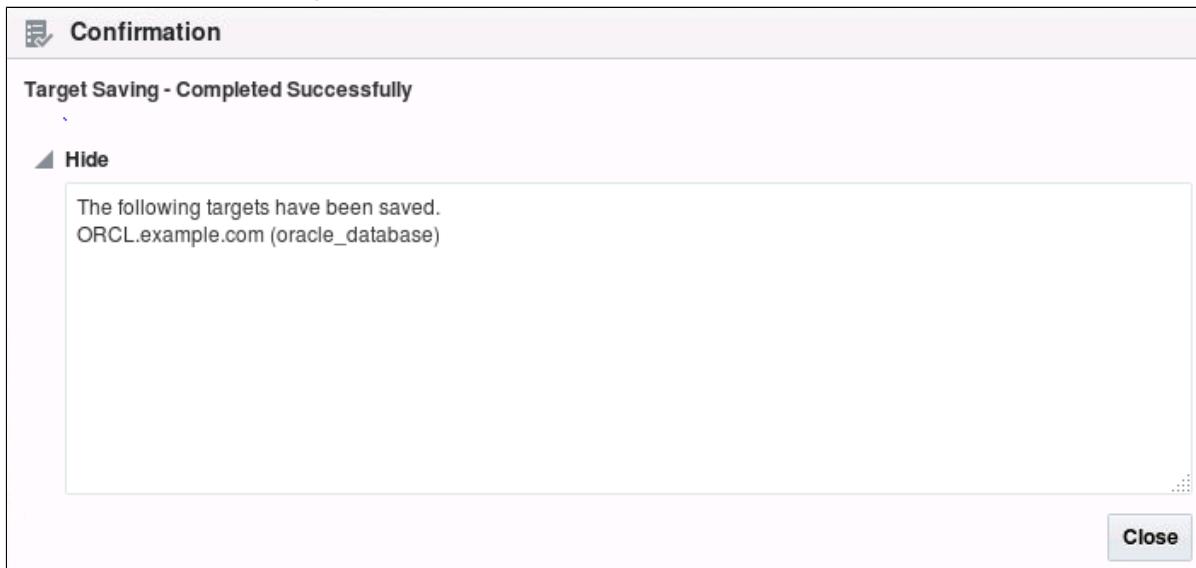
- Select the check box for **ORCL.example.com (Container Database)**.
- In the Role drop-down list, select **SYSDBA**.
- In the Monitor Username box, leave **sys** as is.
- In the Monitor Password box, enter the password for the **SYS** user as specified in Appendix - Product-Specific Credentials.
- Click the **Test Connection** button. An Information dialog box indicates that the connection test was successful. Click **OK** to close that dialog box.



- On the "Database Discovery: Review" page, review the information, and click **Save**.

Target Name	Target Type	Host
ORCL.example.com_sys	Database System	
ORCL.example.com	Database Instance	12cr2db.example.com
Pluggable Database		
ORCL.example.com_CDBROOT	Pluggable Database	12cr2db.example.com
ORCL.example.com_PDB1	Pluggable Database	12cr2db.example.com
ORCL.example.com_PDB2	Pluggable Database	12cr2db.example.com

9. In the Confirmation dialog box, click **Close**.



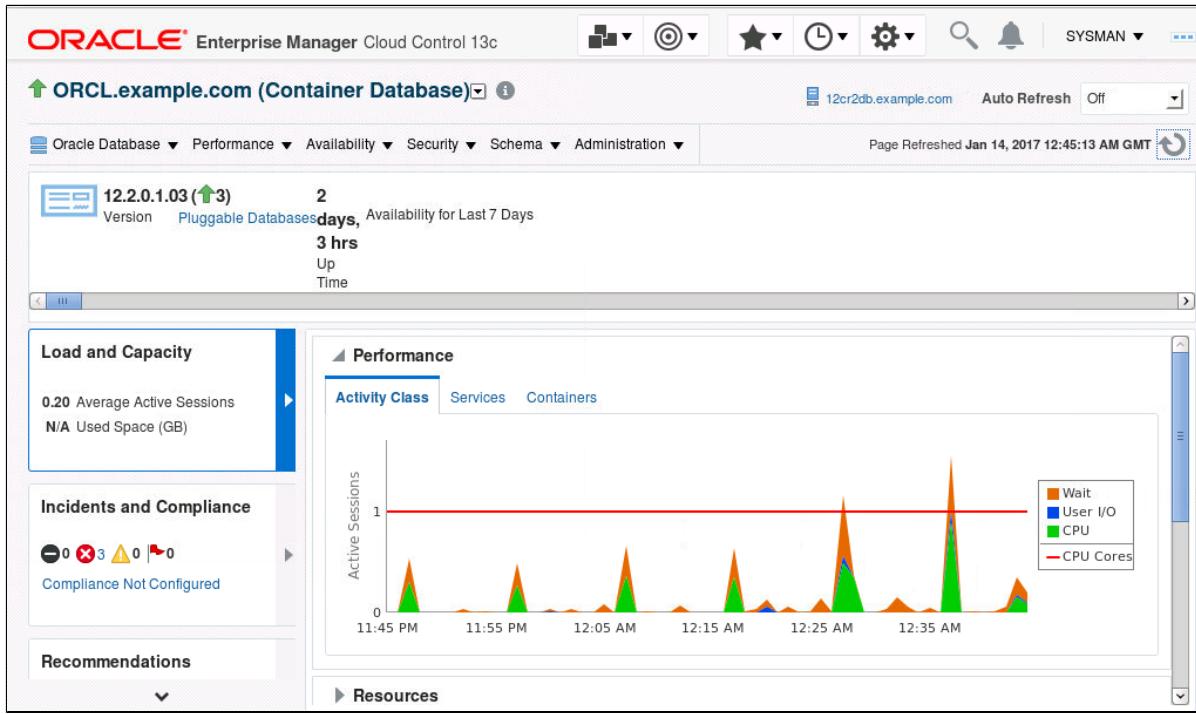
10. In the list of monitored databases, click the **ORCL.example.com** link.

The screenshot shows the "Databases" monitoring interface. At the top, there are dropdown menus for Performance, Availability, Security, Schema, and Administration, and an "Auto Refresh" toggle set to Off. Below that, there are "View" options for Database Load Map and Search List, with "Search List" selected. A search bar with "Find Name" and a magnifying glass icon is present. Below the search bar are buttons for View, Add, Remove, and Configure. The main area is a table with columns: Name, Type, Status, and Target Version. One row is highlighted, showing "ORCL.example.com" as the Name, "Database Instance : Container" as the Type, "Up" as the Status, and "12.2.0.1.0" as the Target Version.

Name	Type	Status	Target Version
ORCL.example.com	Database Instance : Container	Up	12.2.0.1.0

11. Examine the dashboard for the ORCL database. You may need to wait a few minutes for the dashboard to load, and you may need to click the **Refresh** button to refresh the page.

Note: The dashboard may indicate that your pluggable databases are started up.



12. Log out of EM Cloud Control and close the browser window.

Practice 14-2 Using ADDM

Overview

In this practice, you manage the performance of the database instance by using Enterprise Manager Cloud Control (EM Cloud Control).

You could use V\$ views to analyze performance statistics and metrics, but it is much easier through Enterprise Manager Database Express (EM Express) or EM Cloud Control. Whichever tool you use, the key to identifying instance performance issues are wait events and high-cost SQL.

If you want to diagnose the performance decrease in your instance by getting recommendations from a deep analysis, Oracle recommends you use the Automatic Database Diagnostic Monitor (ADDM) through EM Cloud Control. The ADDM feature requires the Enterprise Edition and the Tuning Pack. It is not available in Standard Edition.

Tip:

EM Cloud Control uses data from the Automatic Workload Repository (AWR) to display performance information and initiate database alerts. The *Automatic Workload Repository* (AWR) collects, processes, and maintains performance statistics for problem detection and self-tuning purposes. This gathered data is stored both in memory and in the database, and is displayed in both reports and views. The statistics collected and processed by AWR include:

- Object statistics that determine both access and usage statistics of database segments
- Time model statistics based on time usage for activities, displayed in the V\$SYS_TIME_MODEL and V\$SESS_TIME_MODEL views
- Some of the system and session statistics collected in the V\$SYSSTAT and V\$SESSTAT views
- SQL statements that are producing the highest load on the system, based on criteria such as elapsed time and CPU time
- Active Session History (ASH) statistics, representing the history of recent sessions activity

Snapshots are sets of historical data for specific time periods that are used for performance comparisons by ADDM. By default, Oracle Database automatically generates snapshots of the performance data once every hour and retains the statistics in AWR for eight days. After snapshots are created, ADDM analyzes the data captured in the snapshots to perform its performance analysis.

Assumptions

You are logged in to VM1 as the `oracle` user and in another window you are logged in to VM2 as the `oracle` user.

You completed Practice 14-1 Registering a Database with Enterprise Manager Cloud Control.

Tasks

Start an Application Workload on VM1

1. In the VM1 session, execute the \$HOME/labs/PERF_setup_tuning.sh shell script.

```
$ $HOME/labs/PERF_setup_tuning.sh
...
$
```

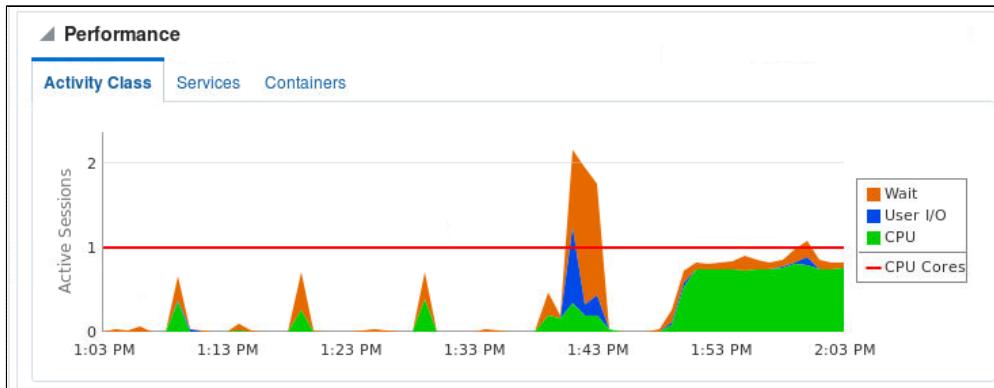
2. Execute the \$HOME/labs/PERF_loop.sh script to start an application workload in PDB1 and PDB2.

```
$ $HOME/labs/PERF_loop.sh
...
$
```

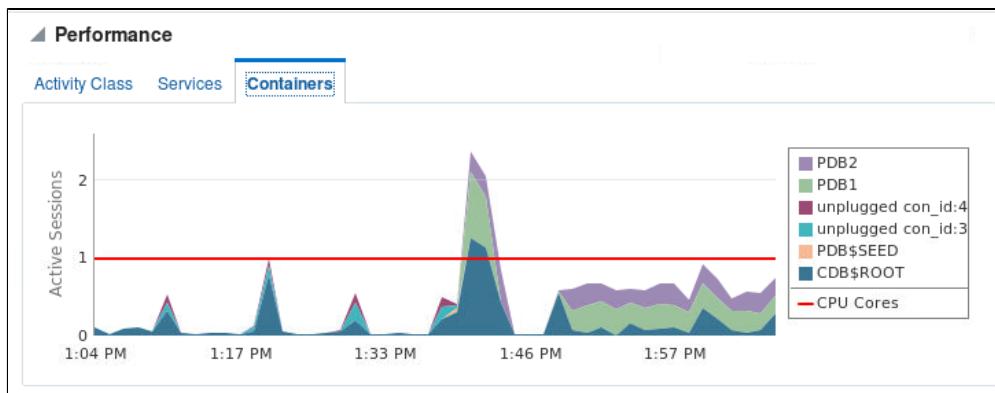
Investigate a Peak Load in Cloud Control on VM2

In this section, you investigate the root cause of the sudden peak load. The ADDM reports and findings identify the performance issues with the highest impact in terms of database time, and provide recommendations based on the experience of Oracle performance consultants and analysis of the root cause of the issue. Even if a recommendation is not provided, the root cause is identified.

1. Start another VNC session and connect to VM2.
2. On VM2, log in to EM Cloud Control.
 - a. Open Firefox.
 - b. If the EM Cloud Control web page is not displayed automatically, enter its URL: <https://localhost:7802/em>.
Note: The format for this URL is https://<machine_name>:<port_number>/em.
 - c. If you get an “Untrusted Connection” message (or something similar depending on the browser and version), add an exception and accept the certificate.
 - d. On the EM Cloud Control login page, enter the user name **SYSMAN**, enter the password, and click **Login**. See [Appendix D - Product-Specific Credentials](#) for the password.
3. View the performance.
 - a. Select **Targets** (bullseye icon in the toolbar), and **Databases**.
 - b. In the list of database instances, click the **ORCL.example.com CDB**.
 - c. On the Activity Class tab of the Performance section, you can see that there is a peak load.



- d. Click the Containers tab to see the performance statistics for each container in the CDB.



4. From the main menu, select **Performance**, and then **Performance Home**.
5. On the Database Login page, enter the following information, and click **Login**.
 - Username: **SYS**
 - Password: See [Appendix - Product-Specific Credentials](#) for the password
 - Role: **SYSDBA**
6. Scroll down to view all the information provided on the Performance Home page.



7. In the Host: Average Runnable Processes area, click the **Run ADDM Now** button.
8. When asked "Are you sure you want to create a new AWR snapshot and run ADDM on this and the previous snapshot?," click **Yes**. This is not the time for the next automatic snapshot to be generated. This explains the

message that asks if you want to create a new snapshot. After the snapshots are created, ADDM analyzes the data captured in the snapshots to perform its performance analysis.

The screenshot shows a confirmation dialog box titled "Confirmation". It contains the message: "Are you sure you want to create a new AWR snapshot and run ADDM on this and the previous snapshot?". At the bottom right are two buttons: "No" and "Yes".

- After you get the message "ADDM has been run successfully.", scroll down to the **ADDM Performance Analysis** section, which displays the task name generated and executed. Notice that the ADDM analysis reports that the major issue is due to SQL statements consuming significant database time.

The screenshot shows the "ADDM Performance Analysis" report. The "Task Name" is "ADDM:1458832693_1_566 (End Time:Jan 14, 2017 2:13:50 PM)". The "Impact (%)" chart shows the following findings:

Impact (%)	Finding	Occurrences (24 hrs ending with analysis period)
65.1	Top SQL Statements	9 of 25
15.6	Session Connect and Disconnect	6 of 25
5	Unusual "Other" Wait Event	6 of 25
2.9	Commits and Rollbacks	7 of 25
2	"Scheduler" Wait Class	1 of 25

- In the Finding column, click **Top SQL Statements**. ADDM found that the culprit SQL statement consumed 94% of the database on CPU and recommends to use SQL Tuning Advisor to improve the SQL execution performance. Your value may be different.

The screenshot shows the "Performance Finding Details: Top SQL Statements" page. It includes sections for Recommendations, Action, Rationale, Findings Path, and a summary table.

Recommendations:

- Schedule SQL Tuning Advisor
- Select All | Select None | Show All Details | Hide All Details
- Select | Details | Category
- ▾ Hide SQL Tuning
- Benefit (%) 39.5

Action:

- Run SQL Tuning Advisor on the SELECT statement with SQL_ID "2bxhpl2vhqnu". [View Tuning History](#) | [Run Advisor Now](#) | [Filters](#)
- SQL Text: `SELECT /*+ monitor USE_NL(d|pi) FULL(d) FULL(pi) FULL(pi) FULL(pi) */ ...`
- SQL ID: `2bxhpl2vhqnu`

Rationale:

- The SQL statement executed in container PDB1 with database ID 2476272392.
- The SQL spent 94% of its database time on CPU, I/O and Cluster waits. This part of database time may be improved by the SQL Tuning Advisor.
- Database time for this SQL was divided as follows: 100% for SQL execution, 0% for parsing, 0% for PL/SQL execution and 0% for Java execution.
- SQL statement with SQL_ID "2bxhpl2vhqnu" was executed 21720 times and had an average elapsed time of 0.0087 seconds.

Findings Path:

- Expand All | Collapse All
- Findings
- SQL statements consuming significant database time were found. These statements offer a good opportunity for performance improvement.

Percentage of Finding's Impact (%)	Additional Information
65.1	

- You can also read the ADDM report to get the whole list of findings and recommendations.
 - Click the browser's **Back** button to return to the ADDM Performance Analysis area.
 - Click the **View Report** button.
 - Read through the report. In this example, the Summary of Findings area of the report states that two recommendations are related to the SQL statement. Other findings relate to session connect and

disconnect, unusual "other" wait events, as well as commits and rollbacks. Your findings may be different.

↑ ORCL.example.com (Container Database) 

Oracle Database ▾ Performance ▾ Availability ▾ Security ▾ Schema ▾ Administration ▾

Advisor Central > Automatic Database Diagnostic Monitor (ADDM):SYS.ADDM:1458832693_1_566 > View Report
View Report

ADDM Report for Task 'ADDM:1458832693_1_566'

Analysis Period

AWR snapshot range from 565 to 566.
Time period starts at 14-JAN-17 02.00.12 PM
Time period ends at 14-JAN-17 02.13.51 PM

Analysis Target

Database 'ORCL' with DB ID 1458832693.
Database version 12.2.0.1.0.
ADDM performed an analysis of instance ORCL, numbered 1 and hosted at 12cr2db.

Activity During the Analysis Period

Total database time was 569 seconds.
The average number of active sessions was .7.

Summary of Findings

Description	Active Sessions Percent of Activity	Recommendations
1 Top SQL Statements	.45 65.12	2
2 Session Connect and Disconnect	.11 15.59	1
3 Unusual "Other" Wait Event	.03 4.96	1
4 Commits and Rollbacks	.02 2.86	1
5 "Scheduler" Wait Class	.01 2	0

12. Log out of EM Cloud Control and close the VNC connections to both VM1 and VM2.



Appendices

ORACLE

Appendix - Product-Specific Credentials

The following table describes user credentials for products that are used in this course.

VM	Product/Application	Username	Password	Notes
VM1	Oracle Database 12.2 - root container	SYS, SYSTEM	oracle_4U	This password is pre-created on your VM.
VM1	Oracle Database 12.2 - PDB1	SYS, SYSTEM, PDBADMIN	oracle_4U	This password is pre-created on your VM.
VM1	Oracle Database 12.2 - PDB1	c##CDB_ADMIN1, PDB1_ADMIN1, INVENTORY, DHAMBY, RPANDYA, and JGOODMAN	oracle_4U	You create the users c##CDB_ADMIN1 and PDB1_ADMIN1 in Practice 5-1 Creating Common and Local Users . You create the INVENTORY user in Practice 5-2 Creating a Local User for an Application . You create the DHAMBY, RPANDYA, and JGOODMAN users in Practice 5-6 Creating Local Users in EM Express . You configure DHAMBY and RPANDYA to change their password at first logon. Change the password to oracle_4U2.
VM1	Oracle Database 12.2 - PDB1	HR	oracle_4U	You unlock this account in Practice 3-2 Hot Cloning a PDB and provide a password .
VM1	Oracle Database 12.2 - PDB1	john	oracle_4U	You create this user in Practice 5-8 Auditing User Activity .
VM1	Oracle Database 12.2 - PDB1	oe, admin_pdb1	oracle_4U	You create these users in Practice 8-1 Managing UNDO Data and Practice 13-2 Avoiding Excessive PGA Memory Usage with the UNDO_setup_tuning.sh script. You create these users in Practice 9-2 Loading Data into a PDB from an External File with the DP_setup.sh script. You require the oe user in Practice 9-4 Unloading External Tables . You create these users in Practice 9-3 Querying External Tables and Practice 10-7 Recovering From an Application Data File Loss with the setup_pdb1.sh script.
VM1	Oracle Database 12.2 - PDB2	admin_pdb2	oracle_4U	You create these users in Practice 8-1 Managing UNDO Data with the UNDO_setup_tuning.sh script.
VM1	Oracle Database 12.2 - PDB1 and PDB2	oetest	oracle_4U	You create this PDB1 user in Practice 9-1 Moving Data From One PDB to Another and Practice 9-2 Loading Data into a PDB from an External File with the DP_setup.sh script.

VM1	Oracle Database 12.2 - PDB1	sh	oracle_4U	You create this user in Practice 9-2 Loading Data into a PDB from an External File with the <code>DP_setup.sh</code> script. You use this user in Practice 9-3 Querying External Tables .
VM1	Oracle Database 12.2	c##test, c##test2	oracle_4U	You create this user in Practice 10-6 Recovering From an Essential Data File Loss .
VM1	Oracle Database 12.2	NGREENBERG, SMAVRIS	oracle_4U	You create these users in Practice 11-2 Resolving Lock Issues with the <code>RESOLVE_LOCK_ISSUES_setup.sh</code> script.
VM1	Enterprise Manager Database Express	SYS	oracle_4U	This password is pre-created on your VM.
VM2	Oracle Database 12.1	SYSTEM	oracle_4U	This password is pre-created on your VM.
VM2	Enterprise Manager Cloud Control	SYSMAN	oracle_4U	This password is pre-created on your VM.

Please ask your instructor for operating system password for VM1 and VM2.

Appendix - Abbreviations

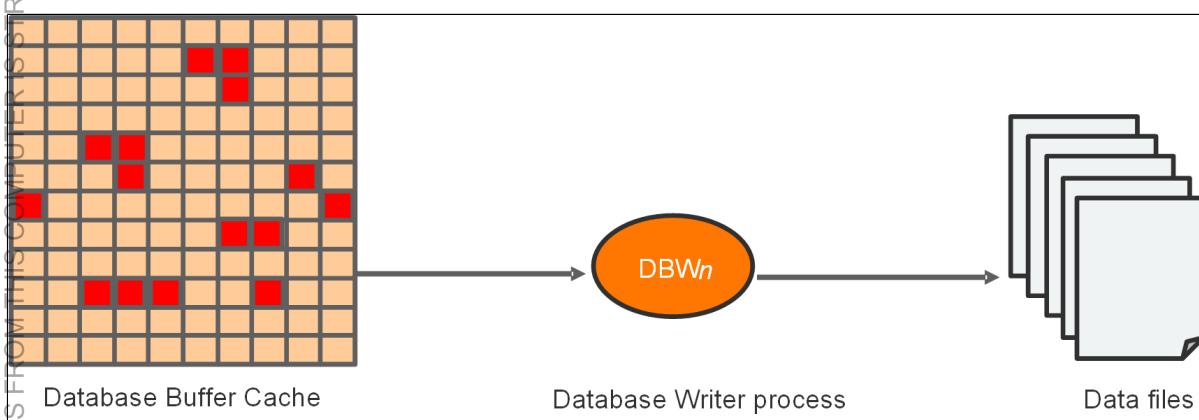
The following table describes abbreviations that are used in this course.

Abbreviation	Description
CDB	container database
non-CDB	non-container database
PDB	pluggable database
VM1	virtual machine 1
VM2	virtual machine 2
SGA	system global area
DBA	database administrator
parameter	initialization parameter
parameter file	initialization parameter file
pfile	text initialization parameter file
spfile	server parameter file
Oracle RAC	Oracle Real Application Clusters

Appendix - Processes

This appendix provides more detail on some of the processes found in the SGA.

Database Writer Process (DBW n)



The Database Writer process (DBW n), as illustrated, writes the contents of buffers to data files. The DBW n processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk. Although one Database Writer process (DBW0) is adequate for most systems, you can configure additional processes to improve write performance if your system modifies data heavily. The additional processes are named DBW1 through DBW9, DBWa through DBWz, and BW36-BW99. These additional DBW n processes are not useful on uniprocessor systems.

When a buffer in the database buffer cache is modified, it is marked dirty and is added to the head of the checkpoint queue that is kept in system change number (SCN) order. This order therefore matches the order of redo that is written to the redo logs for these changed buffers. When the number of available buffers in the buffer cache falls below an internal threshold (to the extent that server processes find it difficult to obtain available buffers), DBW n writes non-frequently used buffers to the data files from the tail of the LRU list so that processes can replace buffers when they need them. DBW n also writes from the tail of the checkpoint queue to keep the checkpoint advancing.

The SGA contains a memory structure that has the redo byte address (RBA) of the position in the redo stream where recovery should begin in case of an instance failure. This structure acts as a pointer into the redo and is written to the control file by the CKPT process once every three seconds. Because the DBW n writes dirty buffers in SCN order, and because the redo is in SCN order, every time DBW n writes dirty buffers from the LRU list, it also advances the pointer held in the SGA memory structure so that instance recovery (if required) begins reading the redo from approximately the correct location and avoids unnecessary I/O. This is known as incremental checkpointing.

Note: There are other cases when DBW n may write (for example, when tablespaces are made read-only or are placed offline). In such cases, no incremental checkpoint occurs because dirty buffers belonging only to the corresponding data files are written to the database unrelated to the SCN order.

The LRU algorithm keeps more frequently accessed blocks in the buffer cache to minimize disk reads. A CACHE option can be placed on tables to help retain blocks even longer in memory.

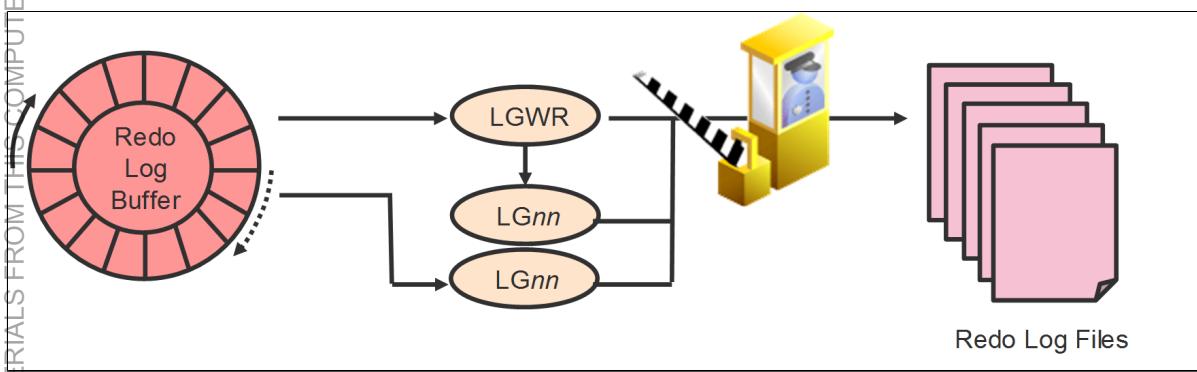
The DB_WRITER_PROCESSES initialization parameter specifies the number of DBW n processes. The maximum number of Database Writer processes is 100. If it is not specified by the user during startup, Oracle Database determines how to set DB_WRITER_PROCESSES based on the number of CPUs and processor groups.

The DBW n process writes dirty buffers to disk under the following conditions:

- When a server process cannot find a clean reusable buffer after scanning a threshold number of buffers, it signals DBW n to write. DBW n writes dirty buffers to disk asynchronously while performing other processing.
- DBW n writes buffers to advance the checkpoint, which is the position in the redo thread (log) from which instance recovery begins. This log position is determined by the oldest dirty buffer in the buffer cache.

In all cases, DBW n performs batched (multiblock) writes to improve efficiency. The number of blocks written in a multiblock write varies by operating system.

Log Writer Process (LGWR)



The Log Writer process (LGWR), as illustrated, is responsible for redo log buffer management by writing the redo log buffer entries to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote. LGWR starts and coordinates multiple helper processes that concurrently perform some of the work. LGWR handles the operations that are very fast, or must be coordinated, and delegates operations to the LGnn that could benefit from concurrent operations, primarily writing the redo from the log buffer to the redo log file and posting the completed write to the foreground process that is waiting.

Because LGnn processes work concurrently and certain operations must be performed in order, LGWR forces ordering so that even if the writes complete out of order, the posting to the foreground processes will be in the correct order.

The redo log buffer is a circular buffer. When LGWR writes redo entries from the redo log buffer to a redo log file, server processes can then copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy. LGWR writes one contiguous portion of the buffer to disk.

LGWR writes:

- When a user process commits a transaction
- When an online redo log switch occurs
- When the redo log buffer is one-third full or contains 1 MB of buffered data
- Before a DBW n process writes modified buffers to disk (if necessary)
- When three seconds have passed since the last write to log files

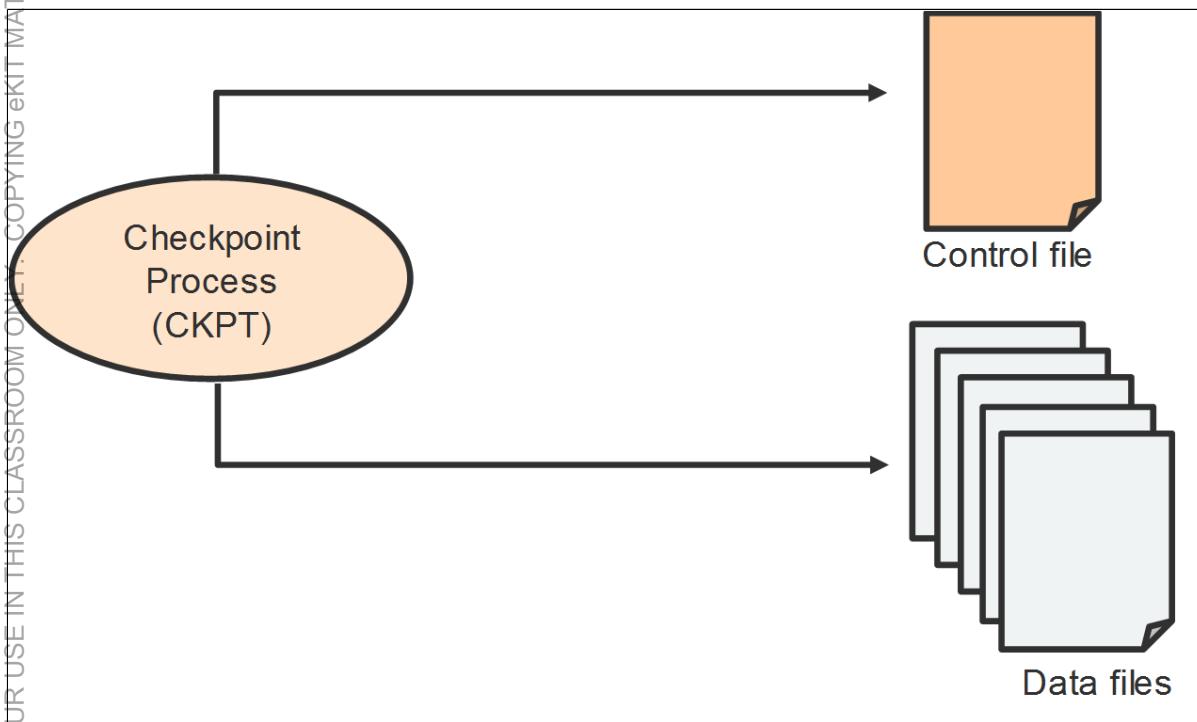
Before DBW n can write a modified buffer, all redo records that are associated with the changes to the buffer must be written to disk (the write-ahead protocol). If DBW n finds that some redo records have not been written, it signals LGWR to write the redo records to disk and waits for LGWR to complete writing the redo log buffer before it can write out the data buffers. LGWR writes to the current log group. If one of the files in the group is damaged or unavailable, LGWR continues writing to other files in the group and logs an error in the LGWR trace file and in the system alert log. If all files in a group are damaged, or if the group is unavailable because it has not been archived, LGWR cannot continue to function.

When a user issues a COMMIT statement, LGWR puts a commit record in the redo log buffer and writes it to disk immediately, along with the transaction's redo entries. The corresponding changes to data blocks are deferred until it is more efficient to write them. This is called a fast commit mechanism. The atomic write of the redo entry containing the transaction's commit record is the single event that determines whether the transaction has committed. Oracle Database returns a success code to the committing transaction, although the data buffers have not yet been written to disk.

If more buffer space is needed, LGWR sometimes writes redo log entries before a transaction is committed. These entries become permanent only if the transaction is later committed. When a user commits a transaction, the transaction is assigned an SCN, which Oracle Database records along with the transaction's redo entries in the redo log. SCNs are recorded in the redo log so that recovery operations can be synchronized in Real Application Clusters and distributed databases.

In times of high activity, LGWR can write to the redo log file by using group commits. For example, suppose that a user commits a transaction. LGWR must write the transaction's redo entries to disk. As this happens, other users issue COMMIT statements. However, LGWR cannot write to the redo log file to commit these transactions until it has completed its previous write operation. After the first transaction's entries are written to the redo log file, the entire list of redo entries of waiting transactions (not yet committed) can be written to disk in one operation, requiring less I/O than do transaction entries handled individually. Therefore, Oracle Database minimizes disk I/O and maximizes performance of LGWR. If requests to commit continue at a high rate, every write (by LGWR) from the redo log buffer can contain multiple commit records.

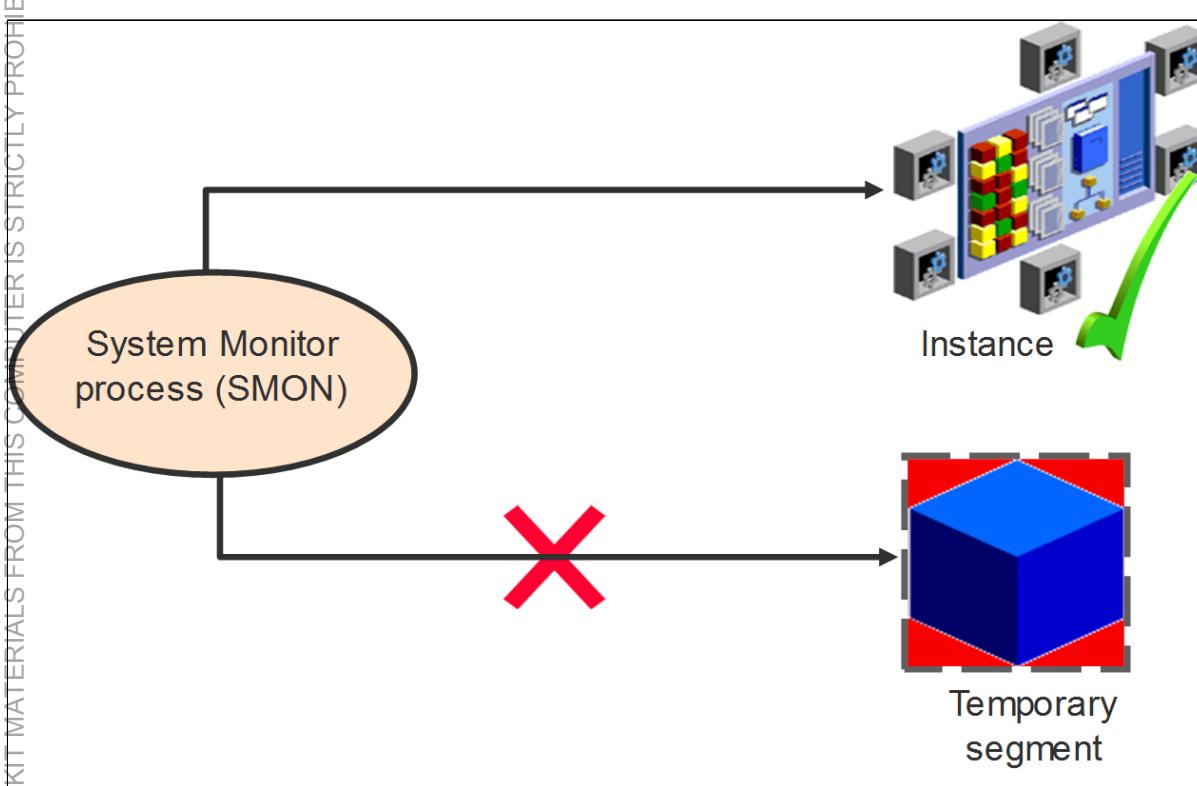
Checkpoint Process (CKPT)



A *checkpoint* is a data structure that defines a system change number (SCN) in the redo thread of a database. Checkpoints are recorded in the control file and in each data file header. They are a crucial element of recovery.

When a checkpoint occurs, Oracle Database must update the headers of all data files to record the details of the checkpoint. This is done by the CKPT process, as illustrated. The CKPT process does not write blocks to disk; DBWn always performs that work. The SCNs recorded in the file headers guarantee that all changes made to database blocks before that SCN have been written to disk.

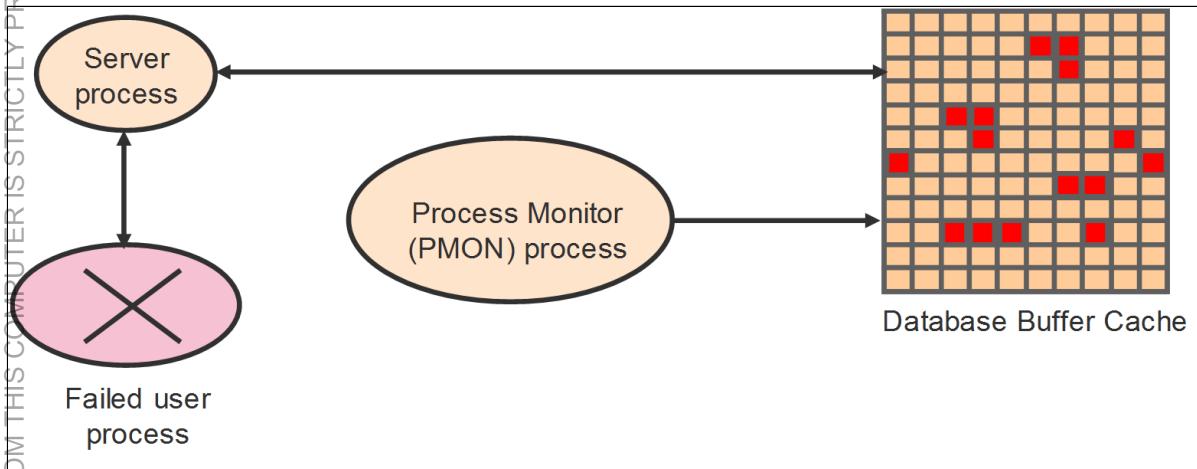
System Monitor Process (SMON)



The System Monitor process (SMON), as illustrated, performs recovery at instance startup if necessary. SMON is also responsible for cleaning up temporary segments that are no longer in use. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online.

SMON checks regularly to see whether the process is needed. Other processes can call SMON if they detect a need for it.

Process Monitor (PMON)

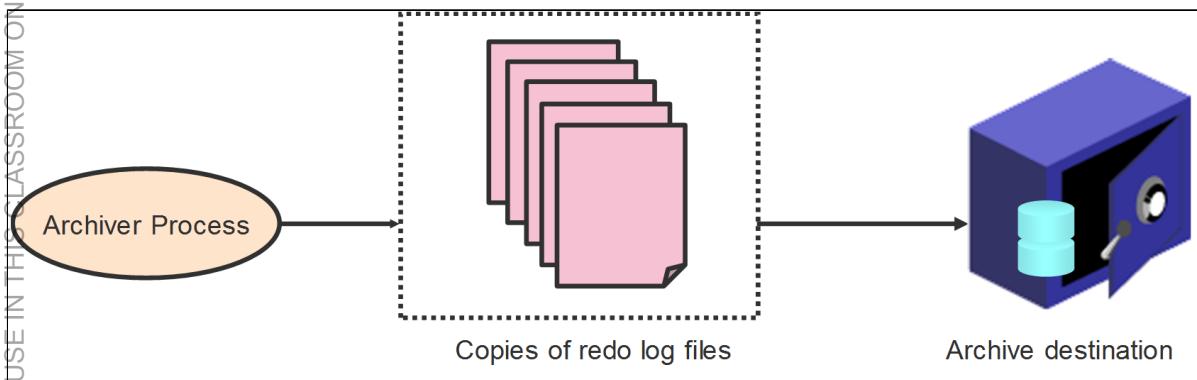


The Process Monitor process (PMON), as illustrated, performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using. For example, it resets the status of the active transaction table, releases locks, and removes the process ID from the list of active processes.

PMON periodically checks the status of dispatcher and server processes, and restarts any that have stopped running (but not any that Oracle Database has terminated intentionally).

Like SMON, PMON checks regularly to see whether it is needed; it can be called if another process detects the need for it.

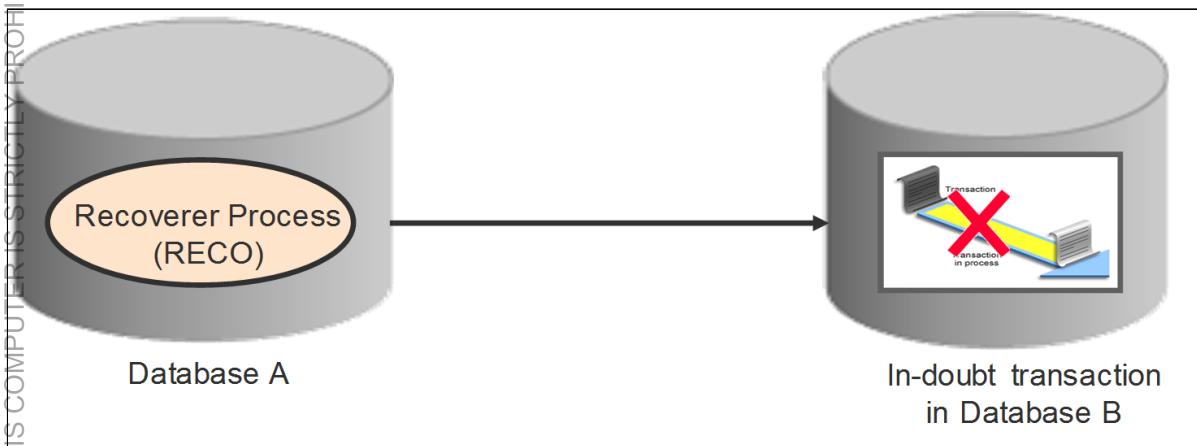
Archiver Processes (ARCn)



The Archiver processes (ARC n), as illustrated, copy redo log files to a designated storage device after a log switch has occurred. ARC n processes are present only when the database is in ARCHIVELOG mode and automatic archiving is enabled.

If you anticipate a heavy workload for archiving (such as during bulk loading of data), you can increase the maximum number of Archiver processes. There can also be multiple archive log destinations. It is recommended that there be at least one Archiver process for each destination. The default is to have four Archiver processes.

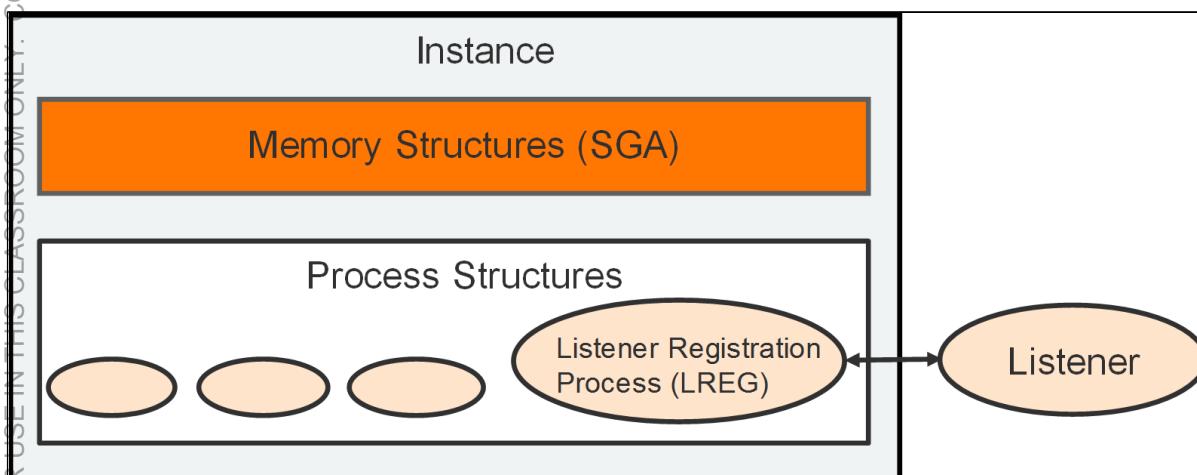
Recoverer Process (RECO)



The Recoverer process (RECO), as illustrated, is a background process that is used with the distributed database configuration that automatically resolves failures involving distributed transactions. The RECO process of an instance automatically connects to other databases involved in an in-doubt distributed transaction. When the RECO process re-establishes a connection between involved database servers, it automatically resolves all in-doubt transactions, removing from each database's pending transaction table any rows that correspond to the resolved in-doubt transactions.

If the RECO process fails to connect with a remote server, RECO automatically tries to connect again after a timed interval. However, RECO waits an increasing amount of time (growing exponentially) before it attempts another connection.

Listener Registration Process (LREG)



The Listener Registration process (LREG), as illustrated, registers information about the database instance and dispatcher processes with the Oracle Net Listener. LREG provides the listener with the following information:

- Names of the database services
- Name of the database instance associated with the services and its current and maximum load
- Service handlers (dispatchers and dedicated servers) available for the instance, including their type, protocol addresses, and current and maximum load

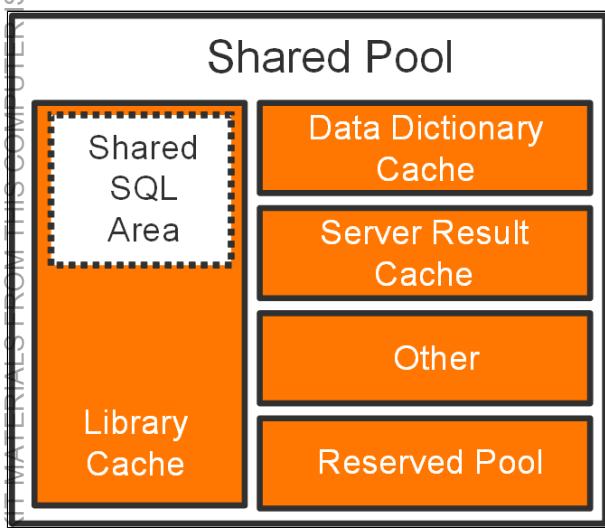
When the instance starts, LREG attempts to connect to the listener. If the listener is running, LREG passes information to it. If the listener is not running, LREG periodically attempts to connect to it. It may take up to 60 seconds for LREG to register the database instance with the listener after the listener has started.

You can use the `ALTER SYSTEM REGISTER` command to immediately initiate service registration after starting the listener.

Appendix - SGA Components

This appendix provides more details about the individual components within the SGA. This content used to be covered in the 12.1 version of this course; however, it is a lot of detail that will be covered later on in the course. Therefore, it is separated out in this appendix and your instructor may or may not cover it with Lesson 1.

Shared Pool



The shared pool portion of the SGA, as illustrated, contains the library cache, the data dictionary cache, the server result cache containing the SQL query result cache and the PL/SQL function result cache, buffers for parallel execution messages, and control structures.

The data dictionary is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database. The data dictionary is accessed so often by Oracle Database that two special locations in memory are designated to hold dictionary data. One area is called the data dictionary cache, also known as the row cache because it holds data as rows instead of buffers (buffers hold entire blocks of data). The other area in memory that holds dictionary data is the library cache. All Oracle Database user processes share these two caches for access to data dictionary information.

Oracle Database represents each SQL statement that it runs with a shared SQL area (as well as a private SQL area kept in the PGA). Oracle Database recognizes when two users are executing the same SQL statement and reuses the shared SQL area for those users. A shared SQL area contains the parse tree and execution plan for a given SQL statement. Oracle Database saves memory by using one shared SQL area for SQL statements run multiple times, which often happens when many users run the same application. When a new SQL statement is parsed, Oracle Database allocates memory from the shared pool to store in the shared SQL area. The size of this memory depends on the complexity of the statement.

Oracle Database processes PL/SQL program units (procedures, functions, packages, anonymous blocks, and database triggers) in the same way it processes individual SQL statements. Oracle Database allocates a shared area to hold the parsed, compiled form of a program unit. Oracle Database allocates a private area to hold values specific to the session that runs the program unit, including local, global, and package variables (also known as package instantiation) and buffers for executing SQL. If more than one user runs the same program unit, then a single, shared area is used by all users, while all users maintain separate copies of their own private SQL areas, holding values specific to their own sessions.

Individual SQL statements contained in a PL/SQL program unit are processed just like other SQL statements. Despite their origins in a PL/SQL program unit, these SQL statements use a shared area to hold their parsed representations and a private area for each session that runs the statement.

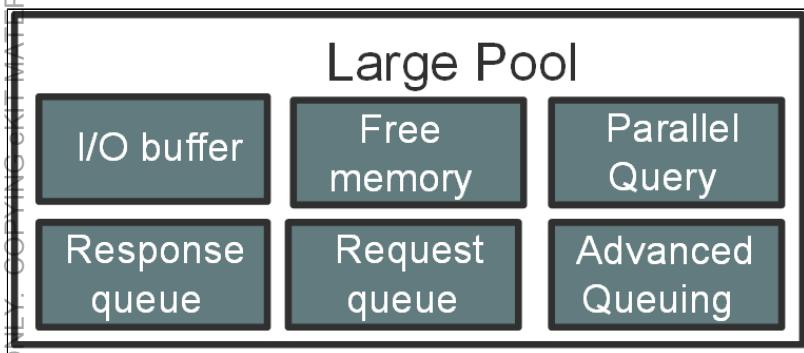
The server result cache contains the SQL query result cache and PL/SQL function result cache, which share the same infrastructure. The result cache contains result sets, not data blocks.

Results of queries and query fragments can be cached in memory in the SQL query result cache. The database server can then use cached results to answer future executions of these queries and query fragments. Because retrieving results from the SQL query result cache is faster than rerunning a query, frequently run queries experience a significant performance improvement when their results are cached.

A PL/SQL function is sometimes used to return the result of a computation whose inputs are one or several parameterized queries issued by the function. In some cases, these queries access data that changes very infrequently compared to the frequency of calling the function. You can include syntax in the source text of a PL/SQL function to request that its results be cached in the PL/SQL function result cache and (to ensure correctness) that the cache be purged when tables in a list of tables experience data manipulation language (DML).

The reserved pool is a memory area in the shared pool that Oracle Database can use to allocate large contiguous chunks of memory.

Large Pool



The database administrator can configure an optional memory area called the large pool, as illustrated, to provide large memory allocations for:

- Session memory for the shared server and the Oracle XA interface (used where transactions interact with multiple databases)
- I/O server processes
- Oracle Database backup and restore operations
- Parallel Query operations
- Advanced Queuing memory table storage

By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle Database can use the shared pool primarily for caching shared SQL and avoid the performance overhead that is caused by shrinking the shared SQL cache.

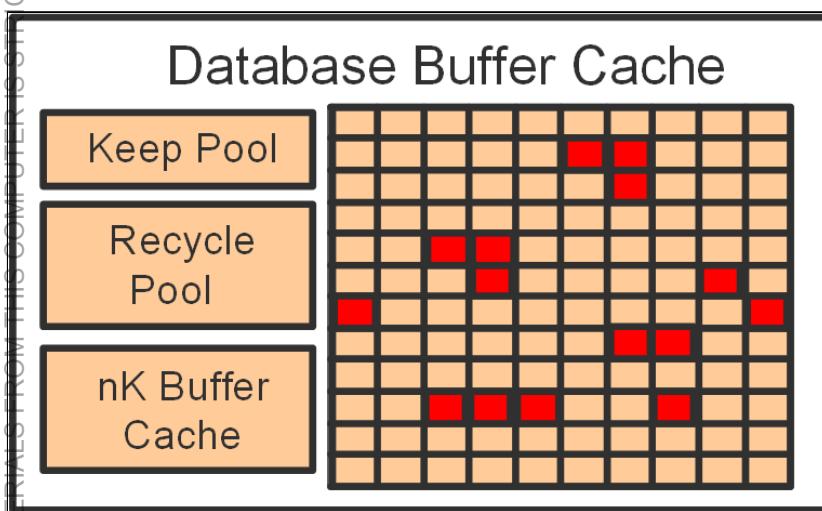
In addition, the memory for Oracle Database backup and restore operations, for I/O server processes, and for parallel buffers is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such large memory requests than the shared pool.

The large pool is not managed by a least recently used (LRU) list.

Java Pool

Java pool memory is used to store all session-specific Java code and data in the Java Virtual Machine (JVM). Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

Database Buffer Cache



The database buffer cache, as illustrated, is the portion of the SGA that holds block images read from the data files or constructed dynamically to satisfy the read consistency model. All users who are concurrently connected to the instance share access to the database buffer cache.

The first time an Oracle Database user process requires a particular piece of data, it searches for the data in the database buffer cache. If the process finds the data already in the cache (a cache hit), it can read the data directly from memory. If the process cannot find the data in the cache (a cache miss), it must copy the data block from a data file on disk into a buffer in the cache before accessing the data. Accessing data through a cache hit is faster than accessing data through a cache miss.

The buffers in the cache are managed by a complex algorithm that uses a combination of least recently used (LRU) lists and touch count. The LRU helps to ensure that the most recently used blocks tend to stay in memory to minimize disk access.

The keep buffer pool and the recycle buffer pool are used for specialized buffer pool tuning. The keep buffer pool is designed to retain buffers in memory longer than the LRU would normally retain them. The recycle buffer pool is designed to flush buffers from memory faster than the LRU normally would.

Additional buffer caches can be configured to hold blocks of a size that is different from the default block size.

Streams Pool

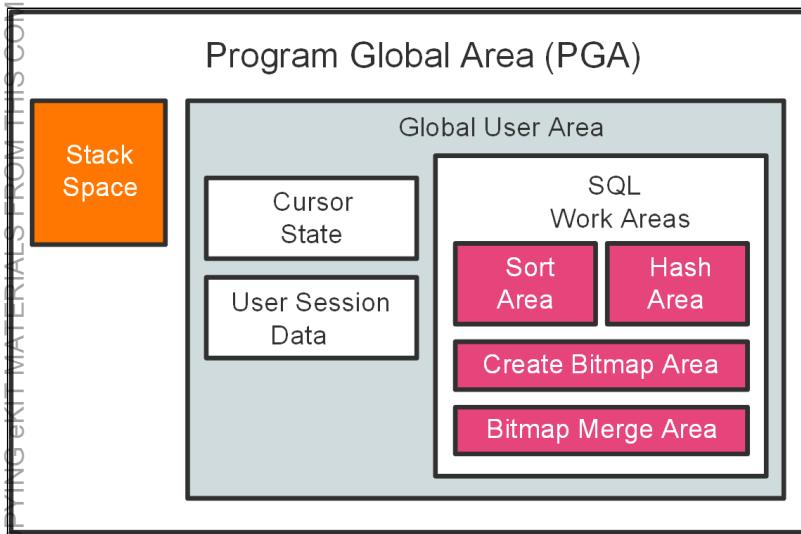
The Streams pool is used by Oracle Streams, Data Pump, and GoldenGate integrated capture and apply processes. The Streams pool stores buffered queue messages, and it provides memory for Oracle Streams capture processes and apply processes. Unless you specifically configure it, the size of the Streams pool starts at zero. The pool size grows dynamically as needed when Oracle Streams is used.

Redo Log Buffer

The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct (or redo) changes that are made to the database by DML, DDL, or internal operations. Redo entries are used for database recovery if necessary.

As the server process makes changes to the buffer cache, redo entries are generated and written to the redo log buffer in the SGA. The redo entries take up continuous sequential space in the buffer. The log writer background process writes the redo log buffer to the active redo log file (or group of files) on disk.

Program Global Area



The Program Global Area (PGA), as illustrated, is a private memory region containing data and control information for a server process. Each server process and background process has a distinct PGA. Access to it is exclusive to that process and it is read only by Oracle code acting on behalf of it. It is not available for developer's code.

Every PGA contains stack space. In a dedicated server environment, each user connecting to the database instance has a separate server process. For this type of connection, the PGA contains a subdivision of memory known as the user global area (UGA). The UGA is composed of the following:

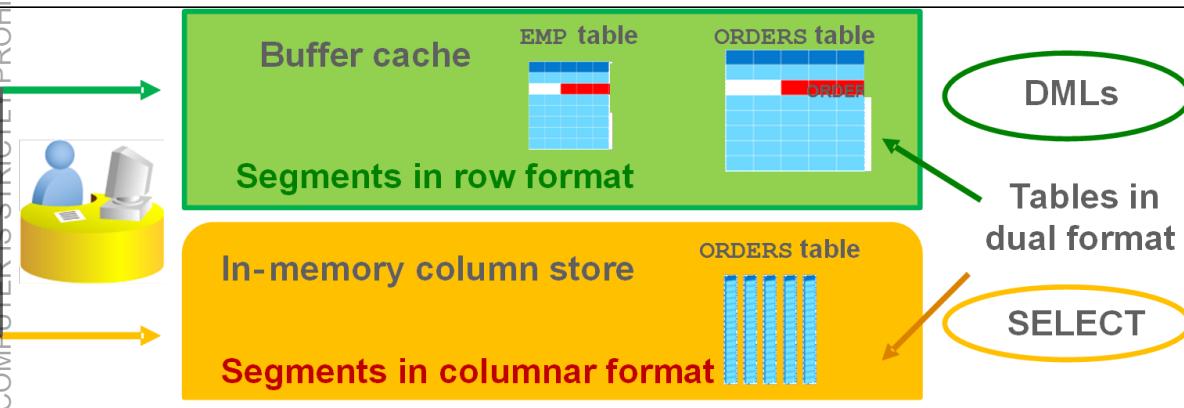
- Cursor area for storing runtime information on cursors
- User session data storage area for control information about a session
- SQL working areas for processing SQL statements consisting of:
 - A sort area for functions that order data, such as ORDER BY and GROUP BY
 - A hash area for performing hash joins of tables
 - A create bitmap area used in bitmap index creation common to data warehouses
 - A bitmap merge area used for resolving bitmap index plan execution

In a shared server environment, multiple client users share the server process. In this model, the UGA is moved into the SGA (shared pool or large pool if configured), leaving the PGA with only stack space.

Fixed SGA

The fixed SGA is an internal housekeeping area containing general information about the state of the database and the instance, and information communicated between processes.

In-Memory Column Store



Note: The In-Memory Column Store feature is included with the Oracle Database In-Memory option.

The In-Memory Column Store feature, as illustrated, enables objects (tables, partitions, and other types) to be stored in memory in a new format known as the columnar format. This format enables scans, joins, and aggregates to perform much faster than the traditional on-disk format, thus providing fast reporting and DML performance for both OLTP and DW environments. This is particularly useful for analytic applications that operate on few columns returning many rows rather than for OLTP that operates on few rows returning many columns. The DBA must define the segments that are to be populated into the in-memory column store (IM column store), such as hot tables, partitions and, more precisely, the more frequently accessed columns.

The in-memory columnar format does not replace the on-disk or buffer cache format. This means that when a segment such as a table or a partition is populated into the IM column store, the on-disk format segment is automatically converted into a columnar format and optionally compressed. The columnar format is a pure in-memory format. There is no columnar format storage on disk. It never causes additional writes to disk and therefore does not require any logging or undo space.

All data is stored on disk in the traditional row format. Moreover, the columnar format of a segment is a transaction-consistent copy of the segment either on disk or in the buffer cache. Transaction consistency between the two pools is maintained.

If sufficient space is allocated to the IM column store in SGA, a query that accesses objects that are populated into the IM column store performs much faster. The improved performance allows more ad hoc analytic queries to be executed directly on real-time transaction data without impacting the existing workload. A lack of IM column store space does not prevent statements from executing against tables that could have been populated into the IM column store.

There are three main advantages:

- Queries run a lot faster: All data can be populated in memory in a compressed columnar format. No index is required and used. Queries run at least 100 times faster than when fetching data from the buffer cache, thanks to the columnar compressed format.
- DMLs are faster: Analytics indexes can be eliminated by being replaced by scans of the IM column store representation of the table.
- Arbitrary ad hoc queries run with good performance, because the table behaves as if all columns are indexed.

The DBA must decide, according to the types of queries and DMLs that are executed against the segments, which segments should be defined as non-in-memory segments and which should be defined as in-memory segments. The DBA can also define more precisely which columns are good candidates for IM column store:

- In row format exclusively: The segments that are being frequently accessed by OLTP-style queries, which operate on few rows returning many columns, are good candidates for the buffer cache. These segments should not necessarily be defined as in-memory segments and should be sent to the buffer cache only.
- In dual format simultaneously: The segments that are being frequently accessed by analytical-style queries, which operate on many rows returning a few columns, are good candidates for IM column store. If a segment is defined as an in-memory segment but has some columns that are defined as non-in-memory columns, the queries that select any non-in-memory columns are sent to the buffer cache and those selecting in-memory columns only are sent to the IM column store. Any fetch-by-rowid is performed on the segment through the buffer cache.

Any DML performed on these objects is executed via the buffer cache.