

Oracle Database 12c: New Features for Administrators

Student Guide - Volume II

D77758GC10

Edition 1.0

May 2013

D80605

ORACLE®

Authors

Dominique Jeunot
Jean-François Verrier

**Technical Contributors
and Reviewers**

James Spiller
Donna Keesling
Maria Billings
Lachlan Williams
Peter Fusek
Dimpi Sarmah
Branislav Valny
Christina Nayagam
Frank Fu
Joel Goodman
Gerlinde Frenzen
Harald Van Breederode
Herbert Bradbury
Hermann Baer
Jim Stenoish
Malareddy Goutam
Patricia McElroy
Paul Needham
Puneet Sangar
Robert McGuirk
Sailaja Pasupuleti
Sean Kim
Sharath Bhujani
Steven Wertheimer
Uwe Hesse
Vimala Jacob

Editor

Smita Kommini

Graphic Designer

Maheshwari Krishnamurthy

Publishers

Giri Venugopal
Michael Sebastian Almeida
Joseph Fernandez

Copyright © 2013, 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.

Contents

I Introduction

- Overview I-2
- Oracle Database Innovation I-3
- Enterprise Cloud Computing I-4
- Oracle Database 12c New and Enhanced Features I-5

1 Enterprise Manager Cloud Control and Other Tools

- Oracle Database 12c New and Enhanced Features 1-2
- Objectives 1-3
- Key Challenges for Administrators 1-4
- Enterprise Manager Cloud Control 1-5
- Cloud Control Components 1-7
- Components and Communication Flow 1-8
- Oracle Management Repository 1-9
- Controlling the Enterprise Manager Cloud Control Framework 1-10
- Starting the Enterprise Manager Cloud Control Framework 1-11
- Stopping the Enterprise Manager Cloud Control Framework 1-12
- Different Target Types 1-13
- Target Discovery 1-14
- Enterprise Manager Cloud Control 1-15
- User Interface 1-16
- Security: Overview 1-17
- Managing Securely with Credentials 1-18
- Distinguishing Credentials 1-19
- Quiz 1-21
- EM Database Express Architecture 1-22
- Configuring Enterprise Manager Database Express 1-23
- Home Page 1-24
- Menus 1-25
- Quiz 1-26
- Database Configuration Assistant 1-27
- Oracle SQL Developer: Connections 1-28
- Oracle SQL Developer: DBA Actions 1-29
- Quiz 1-30
- Summary 1-31

Practice 1 Overview: Using Enterprise Manager Cloud Control 1-32

2 Basics of Multitenant Container Database and Pluggable Databases

Module: Multitenant Container Database and Pluggable Databases 2-1

Oracle Database 12c New and Enhanced Features 2-3

Objectives 2-4

Challenges 2-5

Oracle Database in 11g Release 2 2-6

New Multitenant Architecture: Benefits 2-7

Other Benefits of Multitenant Architecture 2-9

Configurations 2-11

Multitenant Container Database 2-12

Pristine Installation 2-13

Adding User Data 2-14

Separating SYSTEM and User Data 2-15

SYSTEM Objects in the USER Container 2-16

Naming the Containers 2-17

Provisioning a Pluggable Database 2-18

Interacting Within Multitenant Container Database 2-19

Multitenant Container Database Architecture 2-20

Containers 2-21

Questions: Root Versus PDBs 2-22

Questions: PDBs Versus Root 2-23

Terminology 2-24

Common and Local Users 2-25

Common and Local Privileges and Roles 2-26

Shared and Non-Shared Objects 2-27

Data Dictionary Views 2-28

Impacts 2-29

Quiz 2-31

Summary 2-34

Practice 2 Overview: Exploring a Multitenant Container Database 2-35

3 Creating Multitenant Container Databases and Pluggable Databases

Oracle Database 12c New and Enhanced Features 3-2

Objectives 3-3

Goals 3-4

Tools 3-5

Steps to Create a Multitenant Container Database 3-6

Creating a Multitenant Container Database: Using SQL*Plus 3-7

Creating a Multitenant Container Database: Using DBCA 3-9

New Clause: SEED FILE_NAME_CONVERT 3-10
New Clause: ENABLE PLUGGABLE DATABASE 3-11
After CDB Creation: What's New in CDB 3-12
Data Dictionary Views: DBA_xxx 3-13
Data Dictionary Views: CDB_xxx 3-14
Data Dictionary Views: Examples 3-15
Data Dictionary Views: V\$xxx Views 3-16
After CDB Creation: To-Do List 3-17
Automatic Diagnostic Repository 3-18
Automatic Diagnostic Repository: alert.log File 3-19
Quiz 3-20
Practice 3 Overview: Creating a CDB and PDBs 3-22
Provisioning New Pluggable Databases 3-23
Tools 3-24
Method 1: Create New PDB from PDB\$SEED 3-25
Steps: With FILE_NAME_CONVERT 3-26
Steps: Without FILE_NAME_CONVERT 3-27
Method 1: Using SQL Developer 3-28
Synchronization 3-30
Method 2: Plug a Non-CDB into CDB 3-31
Plug a Non-CDB in to CDB Using DBMS_PDB 3-32
Method 3: Clone PDBs 3-33
Method 4: Plug Unplugged PDB in to CDB 3-34
Method 4: Flow 3-35
Plug Sample Schemas PDB: Using DBCA 3-37
Dropping a PDB 3-38
Migrating pre-12.1 Databases to 12.1 CDB 3-39
Quiz 3-40
Summary 3-42
Practice 3 Overview: Creating a CDB and PDBs 3-43

4 Managing Multitenant Container Databases and Pluggable Databases

Oracle Database 12c New and Enhanced Features 4-2
Objectives 4-3
Connection 4-4
Connection with SQL*Developer 4-6
Switching Connections 4-7
Starting Up a CDB Instance 4-8
Mounting a CDB 4-9
Opening a CDB 4-10
Opening a PDB 4-11

| | |
|--|------|
| Closing a PDB | 4-12 |
| Shutting Down a CDB Instance | 4-13 |
| Database Event Triggers: Automatic PDB Opening | 4-14 |
| Changing PDB Mode | 4-16 |
| Changing PDB Mode: With SQL Developer | 4-17 |
| Modifying PDB Settings | 4-18 |
| Instance Parameter Change Impact | 4-19 |
| Instance Parameter Change Impact: Example | 4-20 |
| Quiz | 4-21 |
| Summary | 4-23 |
| Practice 4 Overview: Managing a CDB and PDBs | 4-24 |

5 Managing Tablespaces and Users in CDB and PDBs

| | |
|--|------|
| Oracle Database 12c New and Enhanced Features | 5-2 |
| Objectives | 5-3 |
| Tablespaces in PDBs | 5-4 |
| Creating Permanent Tablespaces in a CDB | 5-5 |
| Assigning Default Tablespaces | 5-6 |
| Creating Local Temporary Tablespaces | 5-7 |
| Assigning Default Temporary Tablespaces | 5-8 |
| Users, Roles, and Privileges | 5-9 |
| Local Users, Roles, and Privileges | 5-10 |
| Creating a Local User | 5-11 |
| Common Users | 5-12 |
| Creating a Common User | 5-13 |
| Common and Local Schemas / Users | 5-14 |
| Common and Local Privileges | 5-15 |
| Granting and Revoking Privileges | 5-16 |
| Creating Common and Local Roles | 5-17 |
| Granting Common or Local Privileges / Roles to Roles | 5-18 |
| Granting Common and Local Roles to Users | 5-19 |
| Granting and Revoking Roles | 5-20 |
| Creating Shared and Non-Shared Objects | 5-21 |
| Restriction on Definer's Rights | 5-22 |
| Quiz | 5-23 |
| Summary | 5-25 |
| Practice 5 Overview: Managing Tablespaces and Users in CDBs and PDBs | 5-26 |

6 Backup, Recovery, and Flashback CDBs and PDBs

| | |
|---|-----|
| Oracle Database 12c New and Enhanced Features | 6-2 |
| Objectives | 6-3 |

| | |
|--|------|
| Goals | 6-4 |
| New Syntax and Clauses in RMAN | 6-5 |
| CDB Backup: Whole CDB Backup | 6-6 |
| CDB Backup: User-Managed Hot CDB Backup | 6-7 |
| CDB Backup: Partial CDB Backup | 6-8 |
| PDB Backup: Whole PDB Backup | 6-9 |
| PDB Backup: Partial PDB Backup | 6-10 |
| Recovery | 6-11 |
| Instance Failure | 6-12 |
| NOARCHIVELOG Mode | 6-13 |
| Media Failure: CDB or PDB Temp File Recovery | 6-14 |
| Media Failure: PDB Temp File Recovery | 6-15 |
| Media Failure: Control File Loss | 6-16 |
| Media Failure: Redo Log File Loss | 6-17 |
| Media Failure: Root SYSTEM or UNDO Data File | 6-18 |
| Media Failure: Root SYSAUX Data File | 6-19 |
| Media Failure: PDB SYSTEM Data File | 6-20 |
| Media Failure: PDB Non-SYSTEM Data File | 6-21 |
| Media Failure: PITR | 6-22 |
| Flashback CDB | 6-24 |
| Special Situations | 6-26 |
| Data Dictionary Views: RC_PDBS | 6-27 |
| Quiz | 6-28 |
| Summary | 6-30 |
| Practice 6 Overview: Managing CDB and PDBs Backup and Recovery | 6-31 |

7 Heat Map, Automatic Data Optimization And Online Data File and Partition Move

| | |
|--|------|
| Automatic Data Optimization and Storage Enhancements | 7-1 |
| Oracle Database 12c New and Enhanced Features | 7-3 |
| Objectives | 7-4 |
| ILM Challenges and Solutions | 7-5 |
| ILM Components | 7-6 |
| ILM Challenges | 7-7 |
| Solutions | 7-8 |
| Components | 7-10 |
| What Is Automatic Data Optimization? | 7-12 |
| Data Classification Levels | 7-13 |
| Heat Map and ADO | 7-14 |
| Enabling Heat Map | 7-15 |
| Monitoring Statistics: Segment-Level | 7-16 |
| DBA_HEAT_MAP_SEGMENT View | 7-17 |

| | |
|--|------|
| Monitoring Statistics: Block Level | 7-18 |
| Monitoring Statistics: Extent Level | 7-19 |
| Defining Automatic Detection Conditions | 7-20 |
| Defining Automatic Actions | 7-21 |
| Compression Levels and Types | 7-22 |
| Creating Compression Policies Tablespace and Group | 7-23 |
| Creating Compression Policies Segment and Row | 7-25 |
| Creating Storage Tiering Policy | 7-26 |
| Storage Tiering: Priority | 7-27 |
| Storage Tiering: READ ONLY | 7-28 |
| Policy Relying on Function | 7-29 |
| Multiple SEGMENT Policies on a Segment | 7-30 |
| Only One Single ROW Policy on a Segment | 7-32 |
| Policy Inheritance | 7-33 |
| Displaying Policies | |
| DBA_ILMPOLICIES/DBA_ILMDATAMOVEMENTPOLICIES | 7-34 |
| Displaying Policies DBA_ILMDATAMOVEMENTPOLICIES | 7-35 |
| Preparing Evaluation and Execution | 7-36 |
| Customizing Evaluation and Execution | 7-37 |
| Monitoring Evaluation and Execution | 7-38 |
| ADO DDL | 7-40 |
| Turning ADO Off and On | 7-41 |
| Stop Activity Tracking and Clean Up Heat Map Statistics | 7-42 |
| Specific Situations of Activity Tracking | 7-43 |
| Quiz | 7-44 |
| Online Move Data File | 7-46 |
| Compression | 7-47 |
| REUSE and KEEP | 7-48 |
| States | 7-49 |
| Compatibilities | 7-50 |
| Flashback Database | 7-51 |
| Online Move Partition | 7-52 |
| Online Move Partition: Benefits | 7-53 |
| Online Move Partition: Compress | 7-54 |
| Quiz | 7-55 |
| Summary | 7-56 |
| Practice 7 Overview: Moving Data Files Online and Practicing ADO | 7-57 |

8 In-Database Archiving and Temporal

| | |
|---|-----|
| Oracle Database 12c New and Enhanced Features | 8-2 |
| Objectives | 8-3 |

| | |
|---|------|
| Archiving Challenges | 8-4 |
| Archiving Solutions | 8-5 |
| In-Database Archiving: HCC | 8-6 |
| Archiving Challenges and Solutions | 8-8 |
| In-Database Archiving | 8-9 |
| ORA_ARCHIVE_STATE column | 8-10 |
| Session Visibility Control | 8-11 |
| Disable Row-Archival | 8-12 |
| Quiz | 8-13 |
| PERIOD FOR Clause Concept | 8-15 |
| Filtering on Valid-Time Columns: Example 1 | 8-16 |
| Filtering on Valid-Time Columns: Example 2 | 8-17 |
| DBMS_FLASHBACK_ARCHIVE | 8-18 |
| Quiz | 8-19 |
| Temporal History Enhancements: FDA Optimization | 8-20 |
| Temporal History Enhancements: User Context Metadata | 8-21 |
| Summary | 8-22 |
| Practice 8 Overview: In-Database Archiving and Temporal | 8-23 |

9 Auditing

| | |
|--|------|
| Module - Security | 9-1 |
| Oracle Database 12c New and Enhanced Features | 9-3 |
| Objectives | 9-4 |
| Types of Auditing | 9-5 |
| Audit Trail Implementation | 9-6 |
| Oracle Database 12c Auditing | 9-8 |
| Security and Performance: Audit Architecture | 9-9 |
| Tolerance Level for Loss of Audit Records | 9-10 |
| Consolidation: Unique Audit Trail | 9-11 |
| Basic Audit Versus Extended Audit Information | 9-12 |
| Extended Audit Information | 9-13 |
| Data Pump Audit Policy | 9-14 |
| Oracle RMAN Audit Information | 9-15 |
| Unified Audit Implementation | 9-16 |
| Quiz | 9-18 |
| Security: Roles | 9-20 |
| Security: SYS Auditing | 9-21 |
| Simplicity: Audit Policy | 9-22 |
| Step 1: Creating the Audit Policy | 9-23 |
| Creating the Audit Policy: Object-Specific Actions | 9-24 |
| Creating the Audit Policy: Condition | 9-25 |

| | |
|---|------|
| Step 2: Enabling / Disabling the Audit Policy | 9-26 |
| Viewing the Audit Policy | 9-27 |
| Using Predefined Audit Policies | 9-28 |
| Including Application Context Data | 9-29 |
| Dropping the Audit Policy | 9-30 |
| Audit Cleanup | 9-31 |
| Quiz | 9-32 |
| Summary | 9-33 |
| Practice 9 Overview: Auditing | 9-34 |

10 Privileges

| | |
|---|-------|
| Oracle Database 12c New and Enhanced Features | 10-2 |
| Objectives | 10-3 |
| Major Challenges | 10-4 |
| Administrative Privileges | 10-5 |
| New Administrative Privileges | 10-6 |
| New Administrative Privilege: SYSBACKUP | 10-7 |
| New Administrative Privilege: SYSDG | 10-8 |
| New Administrative Privilege: SYSKM | 10-9 |
| OS Authentication and OS Groups | 10-10 |
| Password Authentication for SYSBACKUP | 10-12 |
| Password Authentication for SYSDG | 10-14 |
| Oracle Database Vault Data Protection and Administration Privileged Users | 10-15 |
| Privileged Administrators' Auditing | 10-16 |
| Quiz | 10-17 |
| New System Privilege: PURGE DBA_RECYLEBIN | 10-19 |
| Privilege Analysis | 10-20 |
| Privilege Analysis Flow | 10-21 |
| Creating Policies: Database and Role Analysis | 10-22 |
| Creating Policies: Context Analysis | 10-23 |
| Creating Policies: Combined Analysis Types | 10-24 |
| Analyzing and Reporting | 10-25 |
| SYSTEM and OBJECT Used Privileges | 10-26 |
| Used Privileges Results | 10-27 |
| Compare Used and Unused Privileges | 10-28 |
| Views | 10-29 |
| Dropping an Analysis | 10-30 |
| Quiz | 10-31 |
| Privilege Checking During PL/SQL Calls | 10-32 |
| New Privilege Checking During PL/SQL Calls | 10-33 |
| INHERIT (ANY) PRIVILEGES Privileges | 10-34 |

| | |
|--|-------|
| Privilege Checking with New BEQUEATH Views | 10-35 |
| Quiz | 10-36 |
| Summary | 10-38 |
| Practice 10 Overview: Privileges | 10-39 |

11 Oracle Data Redaction

| | |
|---|-------|
| Oracle Database 12c New and Enhanced Features | 11-2 |
| Objectives | 11-3 |
| Oracle Data Redaction: Overview | 11-4 |
| Oracle Data Redaction and Operational Activities | 11-6 |
| Available Redaction Methods | 11-7 |
| Oracle Data Redaction: Examples | 11-8 |
| What Is a Redaction Policy? | 11-9 |
| Managing Redaction Policies | 11-10 |
| Defining a Redaction Policy | 11-11 |
| Adding a Redaction Policy to a Table or View | 11-12 |
| Full Redaction: Examples | 11-13 |
| Partial Redaction: Examples | 11-14 |
| Regular Expression | 11-15 |
| Modifying the Redaction Policy | 11-16 |
| Exempting Users from Redaction Policies | 11-17 |
| Defining Data Redaction Policies by Using Cloud Control 12c | 11-18 |
| Creating a Data Redaction Policy | 11-19 |
| Using Oracle Data Redaction with Other Oracle Database Security Solutions | 11-20 |
| Oracle Database Security Features | 11-21 |
| Best Practices: Preventing Unauthorized Policy Modifications and Exemptions | 11-23 |
| Best Practices: Considerations | 11-24 |
| Summary | 11-25 |
| Practice 11 Overview: Data Redaction | 11-26 |

12 Recovery Manager New Features

| | |
|---|-------|
| Module – High Availability | 12-1 |
| Oracle Database 12c New and Enhanced Features | 12-3 |
| Objectives | 12-4 |
| Separation of DBA Duties | 12-5 |
| Using SQL in RMAN | 12-6 |
| Backing Up and Restoring Very Large Files | 12-7 |
| RMAN Duplication Enhancements | 12-8 |
| Duplicating an Active Database | 12-9 |
| What Is New? | 12-10 |
| The NOOPEN Option | 12-11 |

| | |
|---|-------|
| Duplicating Pluggable Databases | 12-12 |
| Recovering Databases with Third-Party Snapshots | 12-13 |
| Quiz | 12-14 |
| Transporting Data Across Platforms | 12-15 |
| Data Transport | 12-16 |
| Transporting Database: Process Steps - 1 | 12-17 |
| Transporting Database: Process Steps - 2 | 12-18 |
| Transporting Tablespace: Process Steps - 1 | 12-19 |
| Transporting Tablespace: Process Steps - 2 | 12-20 |
| Quiz | 12-21 |
| Table Recovery | 12-22 |
| Recovering Tables from Backups | 12-23 |
| Table Recovery: Graphical Overview | 12-24 |
| Specifying the Recovery Point-in-Time | 12-25 |
| Process Steps of Table Recovery - 1 | 12-26 |
| Customization | 12-27 |
| Quiz | 12-28 |
| Summary | 12-29 |
| Practice 12 Overview: RMAN | 12-30 |

13 Real-Time Database Operation Monitoring

| | |
|--|-------|
| Module – Manageability | 13-1 |
| Oracle Database 12c New and Enhanced Features | 13-3 |
| Objectives | 13-4 |
| Overview | 13-5 |
| Use Cases | 13-6 |
| Current Tools | 13-7 |
| Defining a DB Operation | 13-8 |
| Scope of a Composite DB Operation | 13-9 |
| Database Operation Concepts | 13-10 |
| Identifying a Database Operation | 13-11 |
| Enabling Monitoring of Database Operations | 13-12 |
| Identifying, Starting, and Completing a Database Operation | 13-13 |
| Monitoring the Progress of a Database Operation | 13-14 |
| Monitoring Load Database Operations | 13-15 |
| Monitoring Load Database Operation Details | 13-16 |
| Reporting Database Operations Using Views | 13-17 |
| Reporting Database Operations Using Functions | 13-19 |
| Database Operation Tuning | 13-21 |
| Quiz | 13-22 |
| Summary | 13-24 |

Practice 13 Overview: Monitoring Database Operations 13-25

14 Schema and Data Change Management

Oracle Database 12c New and Enhanced Features 14-2
Objectives 14-3
Database Lifecycle Management Pack New Features 14-4
Change Management Pack Features 14-5
Change Management Pack Components 14-6
Dictionary Baselines 14-7
Dictionary Comparisons 14-9
Dictionary Synchronization 14-11
Comparing Change Propagation and 11g SQL Scripts 14-12
Database Lifecycle Management Pack Schema Change Plans 14-13
Change Requests 14-15
Schema Synchronization 14-17
Database Lifecycle Management Pack Data Comparisons 14-19
DBMS_COMPARISON 14-20
Flow 14-22
Guidelines 14-23
Creating a Data Comparison 14-25
Comparison Job and Results 14-26
Results: Reference Only Rows 14-27
Results: Candidate-Only Rows 14-28
Results: Non-Identical Rows 14-29
Quiz 14-30
Summary 14-32
Practice 14 Overview: Schema Change Plans and Data Comparisons 14-33

15 SQL Tuning Enhancements

Module – Performance 15-1
Oracle Database 12c New and Enhanced Features 15-3
Objectives 15-4
Road Map 15-5
SQL Plan Baseline: Architecture 15-6
SQL Plan Management: Overview 15-8
Adaptive SQL Plan Management 15-9
Automatically Evolving SQL Plan Baseline 15-10
SQL Management Base Enhancements 15-11
Quiz 15-12
Lesson Road Map 15-13
Adaptive Execution Plans 15-14

| | |
|--|-------|
| Dynamic Plans | 15-15 |
| Dynamic Plan: Adaptive Process | 15-16 |
| Dynamic Plans: Example | 15-17 |
| Reoptimization: Cardinality Feedback | 15-18 |
| Cardinality Feedback: Monitoring Query Executions | 15-19 |
| Cardinality Feedback: Reparsing Statements | 15-20 |
| Automatic Re-optimization | 15-21 |
| Quiz | 15-23 |
| Lesson Road Map | 15-24 |
| SQL Plan Directives | 15-25 |
| Using SQL Plan Directives | 15-27 |
| SQL Plan Directives: Example | 15-28 |
| Online Statistics Gathering for Bulk-Load | 15-29 |
| Concurrent Statistics Enhancements in Oracle Database 12c | 15-30 |
| Statistics for Global Temporary Tables | 15-31 |
| Histogram Enhancements | 15-32 |
| Top Frequency Histograms | 15-33 |
| Hybrid Histograms | 15-34 |
| Hybrid Histograms: Example | 15-35 |
| Extended Statistics Enhancements | 15-36 |
| Capturing Column Group Usage | 15-37 |
| Capturing Column Group Usage: Running the Workload | 15-38 |
| Creating Column Groups Detected During Workload Monitoring | 15-40 |
| Automatic Dynamic Sampling | 15-41 |
| Quiz | 15-42 |
| Summary | 15-43 |
| Practice 15 Overview: Using Adaptive Execution Plans and Gathering Statistics Enhancements | 15-44 |

16 Emergency Monitoring, Real-Time ADDM, Compare Period ADDM, and ASH Analytics

| | |
|---|-------|
| Oracle Database 12c New and Enhanced Features | 16-2 |
| Objectives | 16-3 |
| Emergency Monitoring: Challenges | 16-4 |
| Emergency Monitoring: Goals | 16-5 |
| Real-Time ADDM: Challenges | 16-7 |
| Real-Time ADDM: Goals | 16-8 |
| Flow | 16-10 |
| Using the DBMS_ADDM Package | 16-11 |
| Quiz | 16-12 |
| AWR Compare Periods Report | 16-13 |

| | |
|--|-------|
| Tool: Preserved Snapshot Sets | 16-14 |
| Preserved Snapshot Sets | 16-15 |
| AWR Comparison with AWR Snapshot Sets | 16-16 |
| AWR Compare Periods: Examples | 16-17 |
| AWR Comparison with DB Replay | 16-18 |
| What is Missing? | 16-19 |
| Analysis | 16-20 |
| Workload Compatibility | 16-21 |
| Comparison Modes | 16-22 |
| Report: Configuration | 16-23 |
| Report: Finding | 16-24 |
| Report: Resource CPU and I/O | 16-25 |
| Report: Resource Memory | 16-26 |
| Using the DBMS_ADDM Package | 16-27 |
| Quiz | 16-29 |
| ASH: Overview | 16-30 |
| Top Activity Page | 16-31 |
| ASH Analytics Page: Activity | 16-32 |
| Summary | 16-33 |
| Practice 16 Overview: Emergency Monitoring and Compare Period ADDM | 16-34 |

17 Resource Manager and Other Performance Enhancements

| | |
|--|-------|
| Oracle Database 12c New and Enhanced Features | 17-2 |
| Objectives | 17-3 |
| Resource Manager and Pluggable Databases | 17-4 |
| Managing Resources Between PDBs | 17-5 |
| CDB Resource Plan Basics: Share | 17-6 |
| CDB Resource Plan Basics: Limits | 17-9 |
| CDB Resource Plan: Full Example | 17-10 |
| Creating a CDB Resource Plan | 17-11 |
| Creating CDB Resource Plan: SQL Example | 17-12 |
| Viewing CDB Resource Plan Directives | 17-15 |
| Enabling a CDB Resource Plan | 17-16 |
| Maintaining a CDB Resource Plan | 17-17 |
| Managing Resources Within a PDB | 17-18 |
| Managing PDB Resource Plans | 17-19 |
| Putting It Together | 17-20 |
| Considerations | 17-21 |
| Runaway Queries and Resource Manager | 17-22 |
| Default UNIX/Linux Architecture | 17-24 |
| Multi-Process Multi-Threaded UNIX/Linux Architecture | 17-25 |

| | |
|---|-------|
| Multi-Process Multi-Threaded Architecture: Benefits | 17-26 |
| Multi-Process Multi-Threaded Architecture Setup | 17-27 |
| Multi-Process Multi-Threaded Architecture Considerations | 17-28 |
| Multi-Process Multi-Threaded Architecture Monitoring | 17-29 |
| Database Smart Flash Cache Enhancements | 17-30 |
| Enabling and Disabling Flash Devices | 17-31 |
| In-Memory PQ Algorithm: Benefits | 17-32 |
| Smart Flash Cache: New Statistics | 17-33 |
| Temporary Undo: Overview | 17-34 |
| Temporary Undo: Benefits | 17-35 |
| Temporary Undo Setup | 17-36 |
| Temporary Undo Monitoring | 17-37 |
| Summary | 17-38 |
| Practice 17 Overview: Using Resource Manager to Manage Pluggable Database Resources | 17-39 |

18 Tables, Indexes, and Online Operations Enhancements

| | |
|--|-------|
| Oracle Database 12c New and Enhanced Features | 18-2 |
| Objectives | 18-3 |
| Why Multiple Indexes on the Same Set of Columns? | 18-4 |
| Creating Multiple Indexes on the Same Set of Columns | 18-5 |
| Quiz | 18-6 |
| Invisible and Hidden Columns in SQL*Plus | 18-7 |
| SET COLINVISIBLE and DESCRIBE Commands | 18-8 |
| Quiz | 18-9 |
| Online Redefinition: Tables with VPD | 18-10 |
| Online Redefinition: dml_lock_timeout | 18-11 |
| Advanced Row Compression – New Feature Name and Syntax – | 18-12 |
| LOB Compression: New Name | 18-13 |
| Using the Compression Advisor | 18-14 |
| Enhanced Online DDL Capabilities | 18-15 |
| DROP INDEX / CONSTRAINT | 18-16 |
| Index UNUSABLE | 18-17 |
| SET UNUSED Column | 18-18 |
| Summary | 18-19 |
| Practice 18 Overview: Tables and Indexes Enhancements | 18-20 |

19 ADR and Network Enhancements

| | |
|---|------|
| Oracle Database 12c New and Enhanced Features | 19-2 |
| Objectives | 19-3 |
| Automatic Diagnostic Repository | 19-4 |

| | |
|--|-------|
| ADR File Types | 19-5 |
| ADR Files: DDL and DEBUG Log Files | 19-6 |
| ADR Files: Location | 19-7 |
| New ADRCI Command | 19-8 |
| Network Performance: Compression | 19-9 |
| Setting Up Compression | 19-10 |
| SDU Size | 19-11 |
| Setting SDU Size | 19-12 |
| Quiz | 19-13 |
| Summary | 19-14 |
| Practice 19 Overview: ADR Enhancements | 19-15 |

20 Oracle Data Pump, SQL*Loader, and External Tables

| | |
|--|-------|
| Module – Miscellaneous | 20-1 |
| Oracle Database 12c New and Enhanced Features | 20-3 |
| Objectives | 20-4 |
| Full Transportable Export/Import: Overview | 20-5 |
| Full Transportable Export/Import: Usage | 20-6 |
| Full Transportable Export/Import: Example | 20-8 |
| Transporting a Database Over the Network: Example | 20-9 |
| Disabling Logging for Oracle Data Pump Import: Overview | 20-10 |
| Disabling Logging for Oracle Data Pump Import: Usage | 20-11 |
| Disabling Logging for Oracle Data Pump Import: Command-Line Examples | 20-12 |
| Exporting Views as Tables: Overview | 20-13 |
| Exporting Views as Tables: Usage | 20-14 |
| Exporting Views as Tables: Command-Line Examples | 20-15 |
| Specifying the Encryption Password | 20-16 |
| Compressing Tables During Import | 20-17 |
| Creating SecureFile LOBs During Import | 20-18 |
| Quiz | 20-19 |
| SQL*Loader Support for Direct-Path Loading of Identity Columns | 20-20 |
| SQL*Loader and External Table Enhancements | 20-21 |
| SQL*Loader Express Mode | 20-22 |
| Summary | 20-24 |
| Practice 20 Overview: Oracle Data Pump Enhancements | 20-25 |

21 Partitioning Enhancements

| | |
|---|------|
| Oracle Database 12c New and Enhanced Features | 21-2 |
| Objectives | 21-3 |
| Online Partition Operations | 21-4 |
| Online Partition Operations: Benefits | 21-5 |

| | |
|---|-------|
| Online Partition Operation: Compression | 21-6 |
| Enhancements to Reference Partitioning | 21-7 |
| Interval Reference Partitioning | 21-8 |
| TRUNCATE TABLE CASCADE | 21-9 |
| Multi-Partition Maintenance Operations | 21-10 |
| Adding Multiple Partitions | 21-11 |
| Creating a Range-Partitioned Table | 21-12 |
| Adding Multiple Partitions | 21-13 |
| Truncating Multiple Partitions | 21-14 |
| Dropping Multiple Partitions | 21-15 |
| Splitting into Multiple Partitions | 21-16 |
| Splitting into Multiple Partitions: Examples | 21-17 |
| Splitting into Multiple Partitions Rules | 21-18 |
| Splitting into Multiple Partitions: Examples | 21-19 |
| Merging Multiple Range Partitions | 21-20 |
| Merging List and System Partitions | 21-21 |
| Quiz | 21-22 |
| Partitioned Indexes: Review | 21-23 |
| Partial Indexes for Partitioned Tables | 21-24 |
| Partial Index Creation on a Table | 21-25 |
| Specifying the INDEXING Clause at the Partition and Subpartition Levels | 21-26 |
| Creating a Partial Local or Global Index | 21-27 |
| Explain Plan: LOCAL INDEX ROWID | 21-28 |
| Explain Plan: GLOBAL INDEX ROWID | 21-29 |
| Affected Data Dictionary Views Overview | 21-30 |
| Asynchronous Global Index Maintenance | 21-31 |
| The DBMS_PART Package | 21-32 |
| Global Index Maintenance Optimization During Partition Maintenance Operations | 21-33 |
| Forcing an Index Cleanup: Additional Methods | 21-34 |
| Quiz | 21-35 |
| Summary | 21-36 |
| Practice 21 Overview: Partitioning Enhancements | 21-37 |

22 SQL Enhancements and Migration Assistant for Unicode

| | |
|---|------|
| Oracle Database 12c New and Enhanced Features | 22-2 |
| Objectives | 22-3 |
| Increased Length Limits of Data Types | 22-4 |
| Configuring Database for Extended Data Type | 22-5 |
| Using VARCHAR2, NVARCHAR2, and RAW Data Types | 22-6 |
| Database Migration Assistant for Unicode | 22-7 |
| SecureFiles | 22-8 |

SQL Row-Limiting Clause 22-9
SQL Row-Limiting Clause: Examples 22-10
Quiz 22-11
Summary 22-13
Practice 22 Overview: SQL Enhancements 22-14

23 New and Enhanced Features in Other Courses

Further Information 23-2
Suggested Oracle University ILT Courses 23-3

A New Processes, Views, Parameters, Packages, and Privileges

Multitenant Architecture General Architecture Poster A-3
CDB and PDB A-4
ILM: Heat Map and ADO A-6
In-Database Archiving and Temporal Validity A-8
Security: Auditing A-9
Security: Privilege Analysis A-10
Security: Privilege Analysis and New Privileges A-11
Security: Oracle Data Redaction A-12
HA: Flashback Data Archive A-13
Manageability: Database Operations A-14
Manageability: Data Comparisons A-16
Performance: SQL Tuning A-17
Performance: ADDM A-18
Performance: Resource Manager A-19
Performance: Multi-Process Multi-Threaded A-20
Performance: Database Smart Flash Cache A-21
Performance: Temporary UNDO A-22
Performance: Online Operations A-23
Miscellaneous: Partitioning A-24
Miscellaneous: SQL A-25

B Other PDB Creation Methods

Plugging a Non-CDB into CDB Using Data Pump B-2
Plugging a Non-CDB into CDB Using Replication B-3
Cloning PDBs: Using SQL Developer B-4
Plugging Unplugged PDB: Using SQL Developer B-6

11

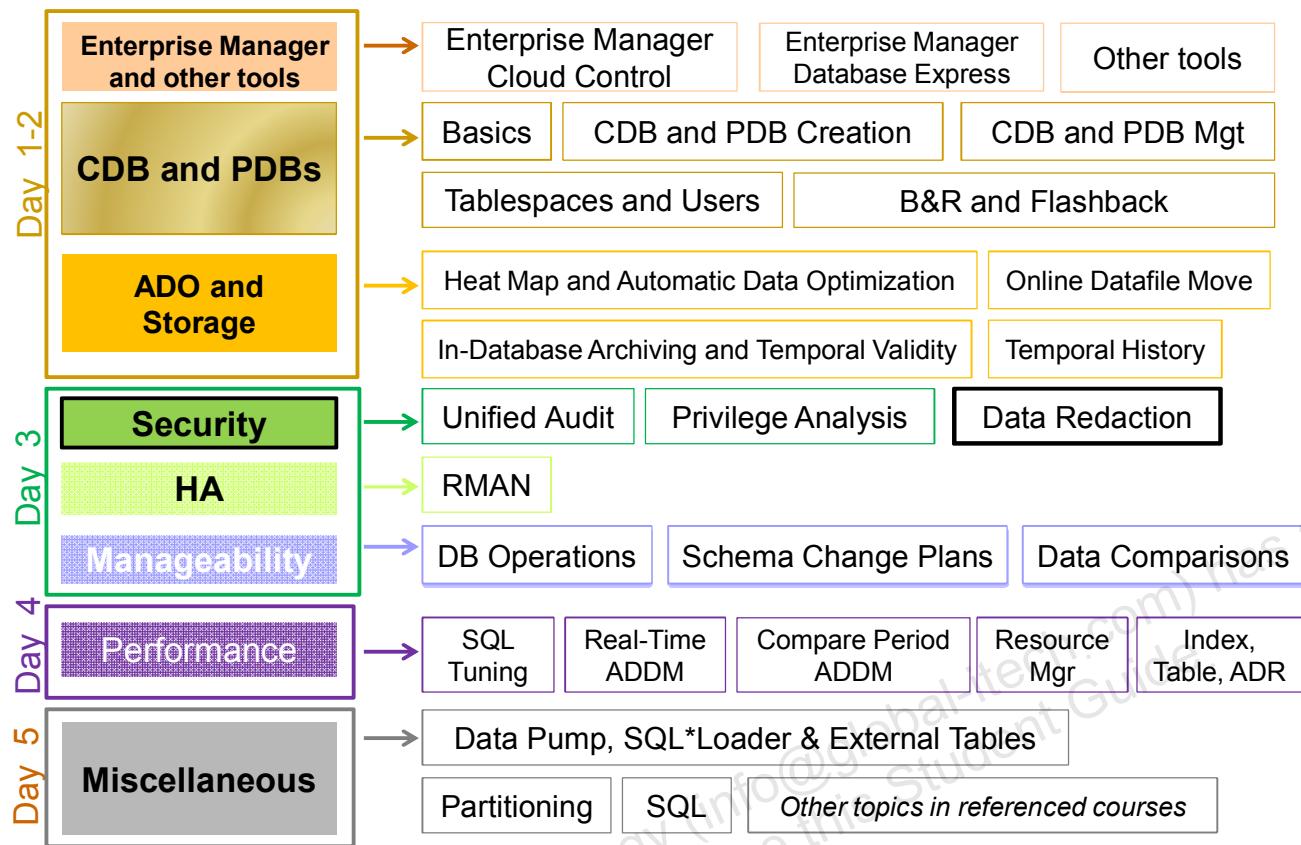
Oracle Data Redaction

A solid red horizontal bar spanning most of the page width.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Data Redaction is a new feature shielding sensitive data from application end users modifying sensitive data columns contained in SQL query results on-the-fly right before the results are returned to applications. Data is not updated and stored in the blocks. **Oracle Data Redaction** is totally different from **Oracle Data Masking**. Oracle Data Masking transforms data by using masking formats and stores new masked data in the blocks.

Objectives

After completing this lesson, you should be able to:

- Describe Oracle Data Redaction
- Manage redaction policies
- View redaction policy information in the data dictionary



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of the Oracle Data Redaction new feature usage, refer to the following guide in the Oracle documentation:

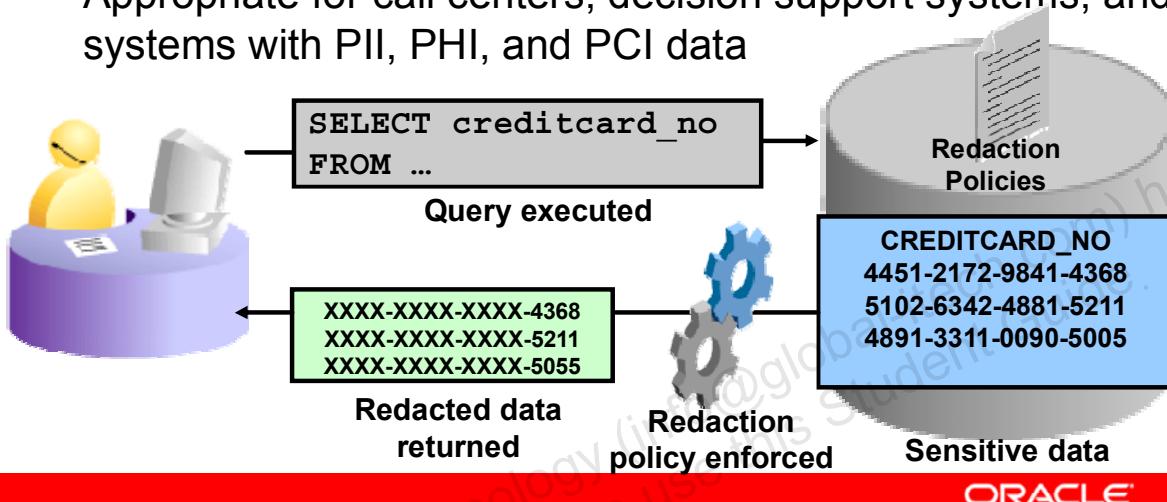
- *Oracle Database Security Guide 12c Release 1 (12.1) – “Using Oracle Data Redaction to Protect Sensitive Data” chapter*
- *Oracle Database 2 Day course – “Limiting Access to Sensitive Data Using Oracle Data Redaction”*
- *Oracle Database PL/SQL Packages and Types Reference 12c Release 1 (12.1) – “DBMS_REDACT” chapter*

Refer to other sources of information available under Oracle Learning Library:

- *Oracle Database 12c New Features Demo Series* demonstrations :
 - *Data Redaction Basics*
- *Oracle By Example (OBE)*:
 - *Protecting Data with Data Redaction*

Oracle Data Redaction: Overview

- On-the-fly redaction based on username, IP address, application context, and other factors
- Transparent, consistent enforcement in the database
- No measurable impact on production workloads
- Appropriate for call centers, decision support systems, and systems with PII, PHI, and PCI data



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Many Oracle Database customers currently prevent the display of sensitive data to end-users by performing redacting in each application. Oracle Data Redaction moves this functionality from the application to the database. This approach has several benefits over redacting data in the application tier, and it is useful in a variety of application scenarios.

Oracle Data Redaction is a transparent, flexible, and simple solution. It modifies sensitive data columns contained in SQL query results on-the-fly right before the results are returned to applications. The columns are redacted according to flexible policies that provide conditional redaction. The policies are managed directly within the database. For maximum transparency, redaction preserves the returned column data type and formatting, and it does not alter the underlying data blocks on disk or in cache. Oracle Data Redaction is designed to be fast so that it can be used on production systems. In addition, it is embedded in the database management system, so no separate installation is required.

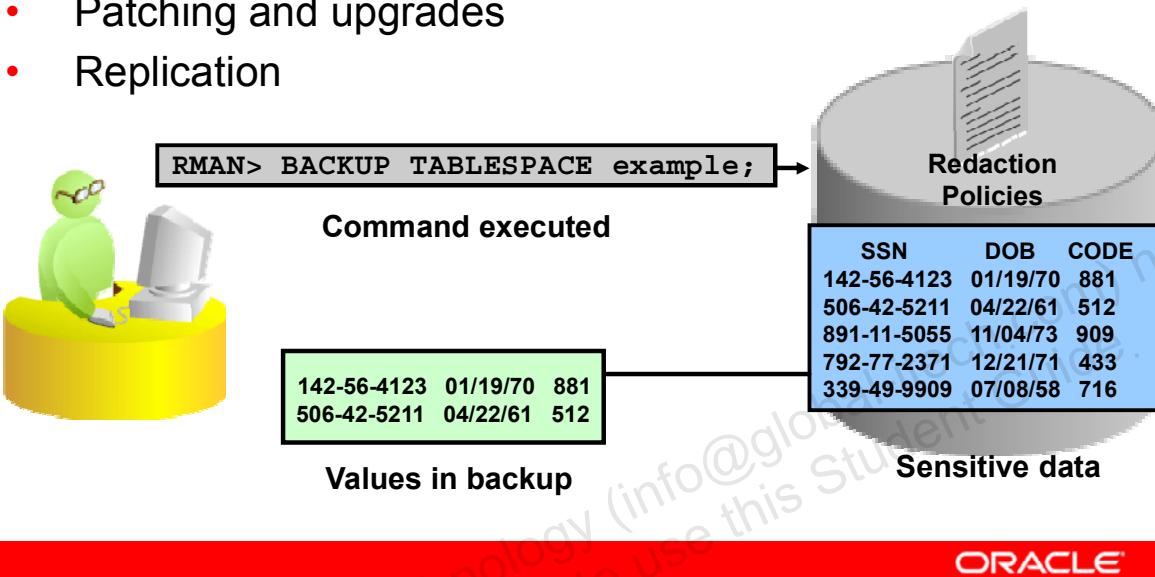
Oracle Data Redaction is useful for many different scenarios. It can be applied to a range of packaged and custom applications to redact application screens, dashboards, and reports. It helps you avoid making code changes in existing call center, human resources, sales, financial, and healthcare applications. These applications frequently manage payment card (PCI), protected health (PHI), or personally identifiable (PII) information that is subject to regulation. Oracle Data Redaction is also useful for decision support systems that aggregate large and diverse sets of data in a single repository for running analytics.

When an application issues a query, data is retrieved from the database and then the redaction policy is applied. Redaction takes place immediately preceding the return of selected data, and only at the top level of the SELECT list.

Oracle Data Redaction and Operational Activities

Operational activities that are not subjected to redaction:

- Backup and restore
- Import and export
- Patching and upgrades
- Replication



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Data retrieved during certain operational activities should not be redacted. Connections as the `SYS` user are always exempt from redaction policies. Utilities such as Data Pump perform recursive SQL as the `SYS` user, so redaction policies are not applied for export and import operations.

In this example, an RMAN command to back up the `EXAMPLE` tablespace is executed. The data is retrieved from the database, but is not redacted in the backup file.

Note: Data Redaction policies associated with tables and views are included in the export and import operation. Therefore, the policies are enabled and the data is redacted when users query the objects in the imported database.

Available Redaction Methods

| Type | Description |
|--------------------|--|
| None | No redaction is performed. |
| Full | Columns are redacted to constant values based on the column data type. |
| Partial | User-specified positions are replaced by a user-specified character. |
| Random | Data type is preserved and different values are output each time. |
| Regular Expression | A “match and replace” is performed based on parameters. |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Data Redaction supports four types of redaction: full, partial, random, and regular expression.

Support for a redaction type of “none” is provided so that you can create redaction policy definitions and evaluate their impact on the application functionality and performance.

Oracle Data Redaction: Examples

Data in the Database -> Redacted Values

Full Redaction

05/24/75 -> 01/01/01

11 Rock Bluff Drive -> xxxxxxxxx

Partial Redaction

068-35-2299 -> ***-**-2299

D1L86YZV8K -> D1*****8K

Regular Expression Redaction

94025-2450 -> 94025-[hidden]

Tom.Lee@acme.com -> [redacted]@acme.com

Random Redaction

4022-5231-5531-9855 -> 4943-6344-0547-0110

09/30/73 -> 11/30/85

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This slide provides examples of data redaction based on the type of redaction that is configured. More detail about defining the redaction type is provided later in the lesson.

What Is a Redaction Policy?

- A redaction policy specifies:
 - What to redact: Specified by a schema, object (table or view), and column
 - How to redact: Specify a redaction method for the column and required parameters for that method
 - When to redact: Specified by a SQL expression that is evaluated for all columns in the table or view and depends on values from:
 - SYS_CONTEXT() for the database environment and context passed by applications
 - XS_SYS_CONTEXT() for Oracle Real Application Security
 - V() and NV() for Oracle Application Express
 - dominates() for Oracle Label Security
- When you create the policy, you can provide only one “how to redact” specification.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To implement Oracle Data Redaction, you define a redaction policy that specifies what should be redacted, when the redaction should take place, and how the redaction should be applied.

First, you identify a schema, table or view, and column to be redacted. Second, you specify the redaction method to use for the column, along with any required parameters. Only one column can be set when you first create the policy, however you can later modify the policy to add additional columns. Last, you specify the exact conditions that must be met for the redaction to be enforced. For example, you can choose to redact the columns by default and show actual data only for certain users or IP addresses. You can join several of these conditions together using logical operators. This “when” condition is set once for the table or view, so it applies to all columns that are added to the policy.

Note that it is good practice to create a white list when defining security policies. A white list enables you to deny by default and add only authorized exceptions to the policy. In the redaction SQL expression, you can insert negative comparisons to build your white list. You can specify that redaction should take place for all users, IP addresses, and so forth that do not match values written into your expression. Black lists, which are the inverse of white lists, also are supported.

Managing Redaction Policies

- You use the procedures in the DBMS_REDACT package to manage redaction policies:
 - ADD_POLICY: Add a redaction policy to a table.
 - DROP_POLICY: Remove a redaction policy from a table.
 - ALTER_POLICY: Change a redaction policy.
 - ENABLE_POLICY: Enable a redaction policy after it is disabled.
 - DISABLE_POLICY: Disable a redaction policy.
- EXECUTE privilege on DBMS_REDACT is required to execute the procedures.
- Enterprise Manager Cloud Control 12c supports Oracle Data Redaction.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You use the DBMS_REDACT PL/SQL package and its procedures to define and manage redaction policies. You can also use Oracle Enterprise Manager Cloud Control 12c to define and manage redaction policies.

To use the procedures in the DBMS_REDACT package, you must be granted the EXECUTE privilege for the package. You do not need any privileges on the tables or views that you specify in the redaction policy.

Best Practice: Restrict the EXECUTE privilege on the DBMS_REDACT package to those users who need to perform data redaction to ensure that policies are not defined or altered inappropriately.

Defining a Redaction Policy

The redaction policy dictates:

- What to redact, as specified by:
 - *Schema name* (OBJECT_SCHEMA)
 - *Object name* (OBJECT_NAME)
 - *Column name* (COLUMN_NAME)
- When to redact, as specified in a *policy expression* (EXPRESSION)
- How to redact, as specified by:
 - *Function type* (FUNCTION_TYPE)
 - *Function parameters* (FUNCTION_PARAMETERS) or *regular expression parameters* (REGEXP_*)



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You define the redaction policy by using the DBMS_REDACT.ADD_POLICY procedure and providing the appropriate parameters, as listed on the slide.

If you do not provide a value for FUNCTION_TYPE, full redaction is used for the specified column.

Restrictions:

- You cannot redact SYS, nor SYSTEM schema objects.
- You cannot redact virtual columns.
- You cannot redact columns of specific data types.
- You can apply VPD policies on other columns than those redacted.

Adding a Redaction Policy to a Table or View

Defining a redaction policy for the SALARY column in the HR.EMPLOYEES table:

```
DBMS_REDACT.ADD_POLICY
(policy_name    => 'EMPSAL_POLICY',
object_schema  => 'HR',
object_name    => 'EMPLOYEES',
column_name    => 'SALARY',
expression     =>
'SYS_CONTEXT(''USERENV'', ''SESSION_USER'') != ''HR-VP''',
function_type  => DBMS_REDACT.FULL);
```



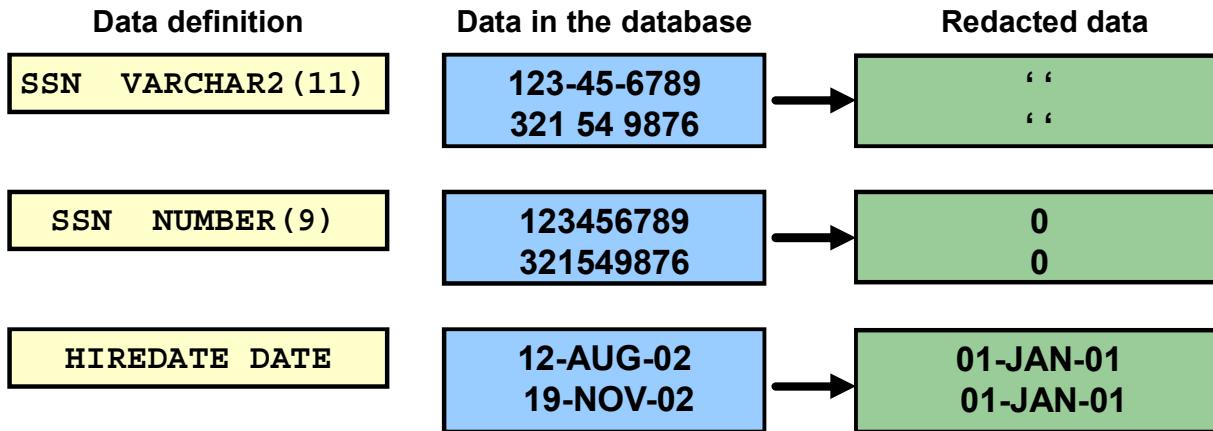
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In this example, a redaction policy named EMPSAL_POLICY is defined on the SALARY column of the HR.EMPLOYEES table. This policy specifies that full redaction is used for the SALARY column values that are returned to all users other than HR-VP.

The REDACTION_POLICIES and REDACTION_COLUMNS views provide respectively the list of redaction policies and redacted columns.

The REDACTION_VALUES_FOR_TYPE_FULL view shows all of the current values for full redaction.

Full Redaction: Examples



Change the default value of full redaction:

```
EXEC DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES
      (varchar_val => 'N');
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

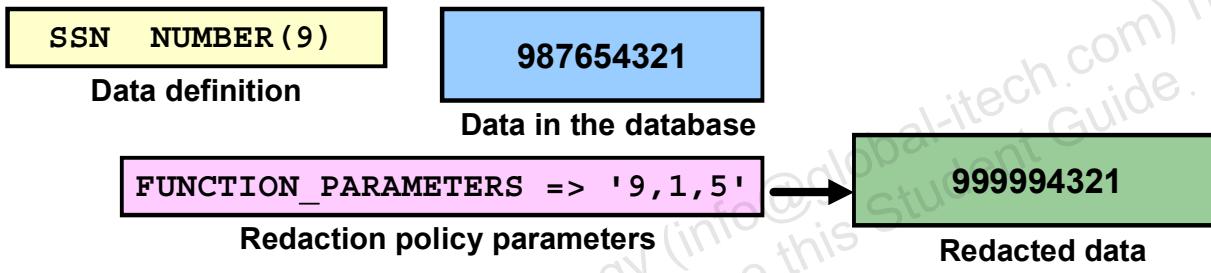
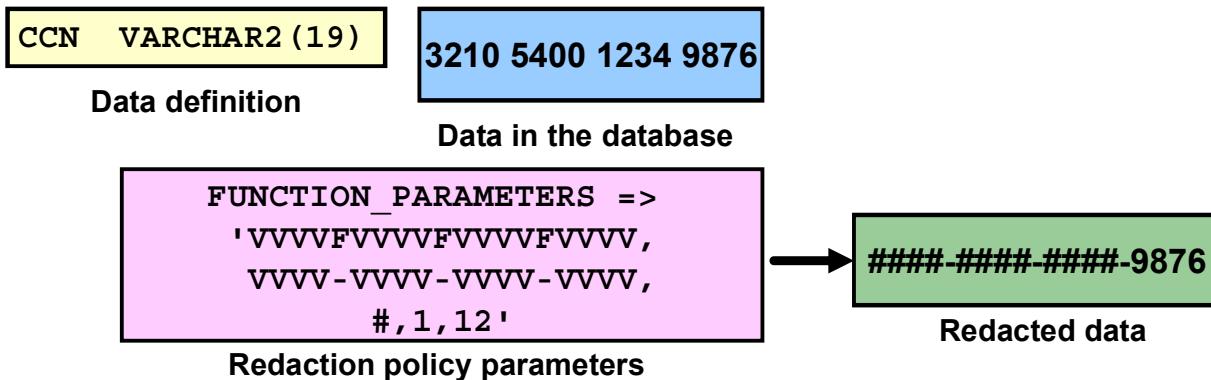
You use full redaction to redact the returned data to a fixed value. As shown in the examples:

- Character data is redacted to a single, blank space.
- Number data is redacted to a single zero.
- Datetime data is redacted to the first of January, 2001.

You specify full redaction by setting FUNCTION_TYPE => DBMS_REDACT.FULL. Full redaction is the default if you do not specify a redaction type in the redaction policy.

Use the UPDATE_FULL_REDACTION_VALUES procedure to modify the default displayed values for full redaction. After you modify a value, you must restart the database for it to take effect. You can find the current values by querying the REDACTION_VALUES_FOR_TYPE_FULL data dictionary view.

Partial Redaction: Examples



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

You specify partial redaction by setting `FUNCTION_TYPE => DBMS_REDACT.PARTIAL`. In addition, you specify parameters so that a portion of the data is redacted and part of the original data is preserved.

The graphic on the slide shows the use of an input parameter that specifies the format characters (F) and values (V) which may be redacted. The output format parameter specifies where to insert formatting characters. The other parameters indicate the redaction character that should be used and the values that should be redacted.

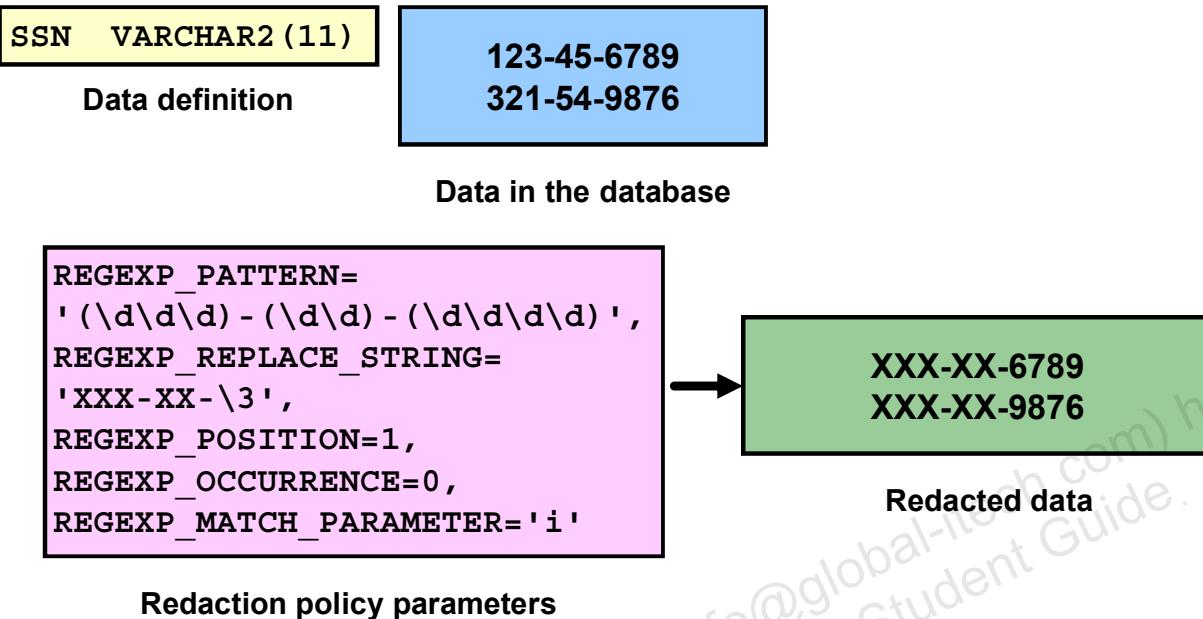
For a number data type, you can specify only numbers 0 through 9 as redaction characters.

For a date data type, lower case letters indicate month, day, year, hour, minute, and second. The value following each letter specifies the redaction value. An uppercase letter indicates that no redaction should be performed for that element.

In the CCN example, the first 12 positions of the VARCHAR2 column are changed to “#” and blanks are replaced with “-”, and the remaining 4 digits left as they are.

In the SSN numeric example, the first 5 positions get numeral 9.

Regular Expression



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You specify regular expression redaction by setting `FUNCTION_TYPE => DBMS_REDACT.REGEXP`.

Use the following parameters to define regular expression redaction:

- Use `REGEXP_PATTERN` to specify a regular expression that represents the column data that will be redacted and that describes the pattern of data which must be matched. In the example on the slide, the `\d` indicates one digit, but not a “d” letter. The whole pattern consists of three digits, then a “-”, then two digits, then a “-”, then four digits.
- Use `REGEXP_REPLACE_STRING` to define how the data should be replaced. In the example on the slide, the `\3` indicates that the data in the third set of parenthesis should be preserved, but the first two sets of digits are replaced by XXX-XX.
- Use `REGEXP_POSITION` to indicate the starting position for the string search.
- Use `REGEXP_OCCURRENCE` to specify how the search and replace operation is to be performed. Use 0 to indicate that all occurrences of the match should be replaced. Use n to indicate that the n th occurrence of the match should be replaced.
- Use `REGEXP_MATCH_PARAMETER` to specify a text literal to change the default matching behavior of the function. The behavior of this parameter is the same as for the `REGEXP_REPLACE` SQL function. Refer to *Oracle Database SQL Language Reference* for detailed information.

Modifying the Redaction Policy

Use the DBMS_REDACT.ALTER_POLICY procedure to alter an existing redaction policy as follows:

- Modify the policy expression.
- Modify the type of redaction for a specified column.
- Modify the function parameters for a specified column.
- Add a column to the redaction policy.
- Remove a column from the redaction policy.

```
DBMS_REDACT.ALTER_POLICY
(policy_name    => 'EMPSAL_POLICY',
 object_schema  => 'HR',
 object_name    => 'EMPLOYEES',
 action         => DBMS_REDACT.MODIFY_EXPRESSION,
 expression     =>
'SYS_CONTEXT(''USERENV'', ''SESSION_USER'') !=''HR-DIR''
) ;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Use the DBMS_REDACT.ALTER_POLICY procedure to modify an existing redaction policy. Specify the type of modification by setting the ACTION parameter. Specify values for EXPRESSION, COLUMN_NAME, and FUNCTION_PARAMETERS or REGEXP_* depending on the value in ACTION.

Use the DBMS_REDACT.ALTER_POLICY procedure to:

- Modify the policy expression by specifying DBMS_REDACT.MODIFY_EXPRESSION for the ACTION parameter. Each policy can have only one policy expression. So when you modify the policy expression, you are replacing the existing policy expression.
- Modify the type of redaction for a specified column by specifying DBMS_REDACT.MODIFY_COLUMN for the ACTION parameter
- Modify the redaction function parameters for a specified column by specifying DBMS_REDACT.MODIFY_COLUMN for the ACTION parameter
- Add a column to the redaction policy by specifying DBMS_REDACT.ADD_COLUMN for the ACTION parameter. You must also specify the redaction type and any required parameters.
- Remove a column from the redaction policy by specifying DBMS_REDACT.DROP_COLUMN for the ACTION parameter

Exempting Users from Redaction Policies

- Conditions included in policy expressions may allow users to see actual data.
- SYS is exempt from all redaction policies.
- Grant the EXEMPT REDACTION POLICY system privilege to exempt other users from all redaction policies.
- Best Practices:
 - Use default deny (white list) conditions in policy expressions.
 - Grant the EXEMPT REDACTION POLICY privilege judiciously to ensure that the redaction policies are enforced appropriately.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SYS user is exempt from redaction policies and is able to view actual values for data.

If other users need access to the actual values, you must grant them the EXEMPT REDACTION POLICY system privilege. This privilege exempts the users from all redaction policies.

Users granted the DBA role are also exempt from redaction policies because the DBA role contains the EXP_FULL_DATABASE role, which is granted the EXEMPT REDACTION POLICY system privilege.

Because applications may need to perform CREATE TABLE AS SELECT operations that involve redacted source columns, you can grant the application the EXEMPT DDL REDACTION POLICY system privilege.

Defining Data Redaction Policies by Using Cloud Control 12c

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The top navigation bar includes links for Enterprise, Targets, Favorites, History, Setup, Help, SYSMAN, and Log Out. The main menu on the left is set to 'database'. The current page is 'Data Redaction'. A yellow callout box points to the 'Create' button in the search bar, which contains fields for Schema, Table/View, and Policy Name, followed by a 'Go' button. Below this is a table titled 'Data Redaction Policies' with columns: Schema, Table/View, Policy Name, Enabled, and Masked Columns. The 'Format' dropdown menu is open, showing options like Create, Edit, View, Enable, Disable, Delete, and Detach. The status bar at the bottom indicates the user is logged in as SYSTEM and the page was refreshed on June 6, 2012, at 9:03:19 PM PDT.

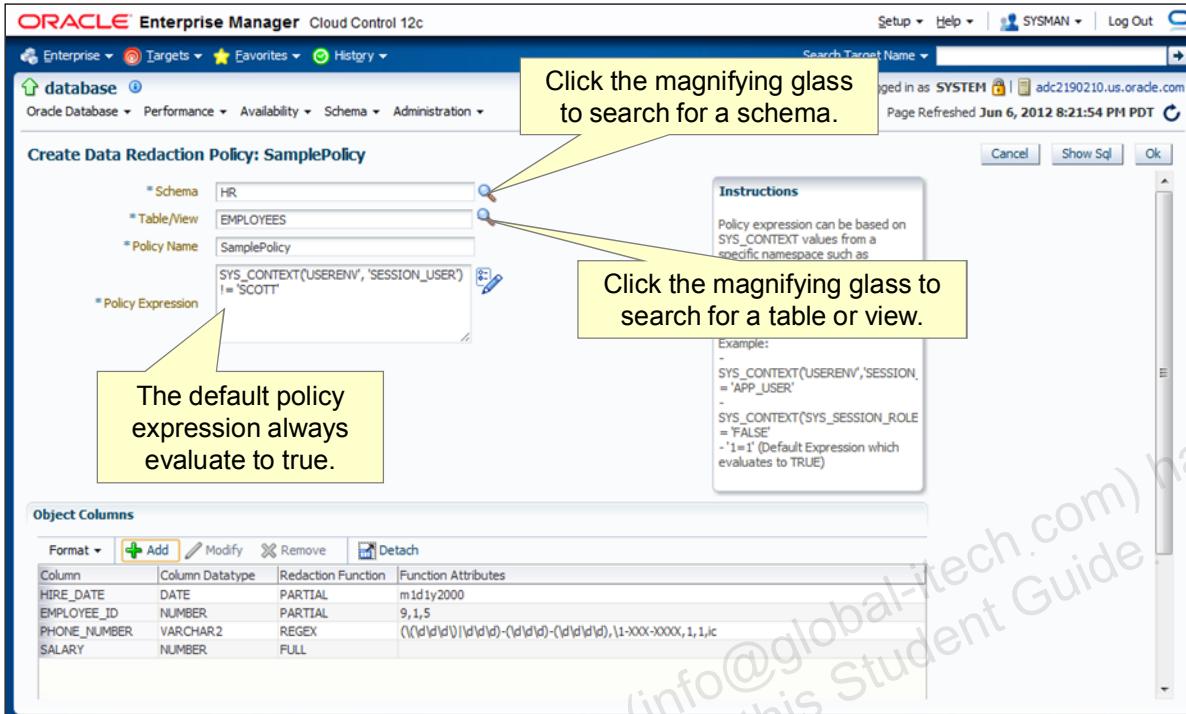
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Enterprise Manager Cloud Control 12c provides a graphical user interface that enables you to define data redaction policies. You access the Data Redaction page by selecting Administration -> Security -> Oracle Data Redaction.

To define a new redaction policy, click Create.

Creating a Data Redaction Policy



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

On the Create Data Redaction Policy page, specify the schema and table that the redaction policy applies to. You can click the magnifying glass next to the Schema and Table/View fields to open a search window. Provide a name for the policy and specify the policy expression. By default, policy expression is set to 1=1, which evaluates to true meaning that redaction will always take place.

After specifying the schema and table or view name, add columns to the redaction policy. The example in the slide specifies the HIRE_DATE column of the HR.EMPLOYEES table. For this column, partial redaction is indicated and the redacting format is specified for a DATE data type. Predefined redaction templates are available in Oracle Enterprise Manager to enable you to easily specify redaction for common types of sensitive data. In this example, the “Date to Millennium” redaction template is selected. This template sets the redaction format to m1d1y2000. Date values will be redacted to 01-JAN-00, the first day of the millennium. Note that you can manually change the template’s default value. You can select the Custom template if you want to manually enter all values.

Add the EMPLOYEE_ID column to the redaction policy. Again, partial redaction is indicated with the required function attributes for a column of NUMBER data type.

Specify the policy expression so that redaction is in effect when the session user is not the SCOTT user.

Finally create the redaction policy by clicking OK. A message shows that the redaction policy was successfully created and enabled by default.

Using Oracle Data Redaction with Other Oracle Database Security Solutions

- Serves different use cases
- Is complementary to other Oracle Database security solutions and can be used with them in tandem

| | |
|---|--|
| ✓ | Oracle Advanced Security Transparent Data Encryption (TDE) |
| ✓ | Oracle Database Vault |
| ✓ | Oracle Database Firewall |
| ✓ | Oracle Audit Vault |
| ✓ | Oracle Label Security (OLS) |
| ✓ | Oracle Virtual Private Database (VPD) |
| ✓ | Oracle Enterprise Manager Data Masking Pack |
| ✓ | Oracle Real Application Security (RAS) |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Data Redaction serves different use cases than other Oracle Database security solutions and should be viewed as complementary to them. Oracle Database security offerings can be paired with Oracle Data Redaction to provide a more secure environment.

Oracle Database Security Features

| Oracle Database Security Product or Feature | Description |
|--|---|
| Oracle Data Masking Pack | Physically alters and masks production data before you distribute copies of it to development and test environments |
| Oracle Virtual Private Database | Dynamically rewrites the SQL query to filter the data returned Can prevent certain rows from being returned, but for columns, it is limited to setting them to NULL. |
| Oracle Advanced Security Transparent Data Encryption (TDE) | Transparently encrypts and decrypts data in storage without impacting applications Does not alter the SQL query results Requires managing the encryption keys outside of the database |
| Real Application Security | Powerful fine-grained features for controlling the data returned Requires uptake of the programming framework and custom coding Suitable for applications where you are able to modify the code |
| Oracle Database Vault | Applies an additional layer of access control to highly privileged database user accounts Disallows selected operations and returns an error code |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Data Masking Pack is an optional feature of Oracle Enterprise Manager Cloud Control 12c. By irreversibly replacing original, sensitive data with fictitious data, it enables customers to safely share data with IT developers and business partners. It provides end-to-end secure automation for provisioning test databases from production in compliance with regulations.

You use Oracle Virtual Private Database (VPD) to create security policies that control database access at the row and column level. Essentially, VPD adds a dynamic WHERE clause to a SQL statement that is issued against the table, view, or synonym to which a VPD security policy was applied.

You can use VPD to redact column data, but only to change columns to NULL, which may cause errors in applications.

Oracle Data Redaction and VPD are complementary features that you can use together. For example, you can use VPD to filter the rows that are returned by the query, and you can use Oracle Data Redaction to redact the columns that are displayed.

Transparent Data Encryption (TDE) is a feature of Oracle Advanced Security. You use TDE to encrypt sensitive data that is stored in data files. It uses a TDE master encryption key that is stored in an Oracle Wallet or a Hardware Security Module (HSM).

For detailed information on TDE, refer to *Oracle Database Advanced Security Administrator's Guide 12c Release 1*.

Oracle Data Redaction and TDE are complementary features. They can be used together to protect the data that is stored in the database and to mask the data when it is returned to a query.

Oracle Database Real Application Security is a new feature of Oracle Database 12c. It is a database authorization model that supports declarative security policies and enables end-to-end security for multilayer applications. It provides an integrated solution for securing the database and application user communities. Also, it advances the security architecture of Oracle Database to meet existing and emerging demands of applications that are developed for the Internet.

For additional information about this feature, refer to the *Oracle Database 12c: Real Application Security* self-study course.

Additional information about Oracle Database Vault follows.

Best Practices: Preventing Unauthorized Policy Modifications and Exemptions

- Limit the EXECUTE privilege on the DBMS_REDACT package.
- Limit the EXEMPT REDACTION POLICY privilege.
- If you use Oracle Database Vault to restrict privileged user access, you can create realms to:
 - Further limit the execution of the DBMS_REDACT package and granting of the EXEMPT REDACTION POLICY privilege
 - Limit access to the REDACTION_POLICIES and REDACTION_COLUMNS views



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To prevent unauthorized policy modifications and exemptions, limit grants of the EXECUTE privilege on the DBMS_REDACT package and granting of the EXEMPT REDACTION POLICY privilege.

Oracle Database Vault restricts access to certain areas of the database even for highly privileged users. If you use Database Vault, you can create realms to further limit execution of the DBMS_REDACT package and granting of the EXEMPT REDACTION POLICY privilege. You can also use Database Vault to restrict access to the REDACTION_POLICIES and REDACTION_COLUMNS views.

For additional information on Oracle Database Vault, refer to the *Implementing Oracle Database Vault* course and *Oracle Database Vault Administrator's Guide 12c Release 1*. Information on new features in Oracle Database Vault in Oracle Database 12c is available in the Oracle Database Vault lesson in this self-study course.

Best Practices: Considerations

- Choose the columns to redact selectively; redact only what is needed.
- Keep the policy expression logic as simple as possible.
- When defining regular expression policies, keep the regular expressions simple.
- Partial and full redaction policies generally have higher performance than regular expression policies which must be compiled each time they are used.
- You cannot perform a CTAS operation where any of the columns being selected is protected.
- You can apply only one policy on a table or view.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To maximize performance, redact only the columns that are needed and keep the logic in policy expressions and regular expressions as simple as possible. Note that per-row processing time (CPU time) increases as more columns are redacted in a given table or view. Also, the evaluation of policy expressions occurs once for each query execution, so applications that repeat queries to fetch one row of data at a time (rather than fetching multiple rows at once) may detect overhead due to redaction.

Summary

In this lesson, you should have learned how to:

- Describe Oracle Data Redaction
- Manage redaction policies
- View redaction policy information in the data dictionary

Practice 11 Overview: Data Redaction

These practices cover the following topics:

- Using full data redaction
- Using partial data redaction



Module

High Availability



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

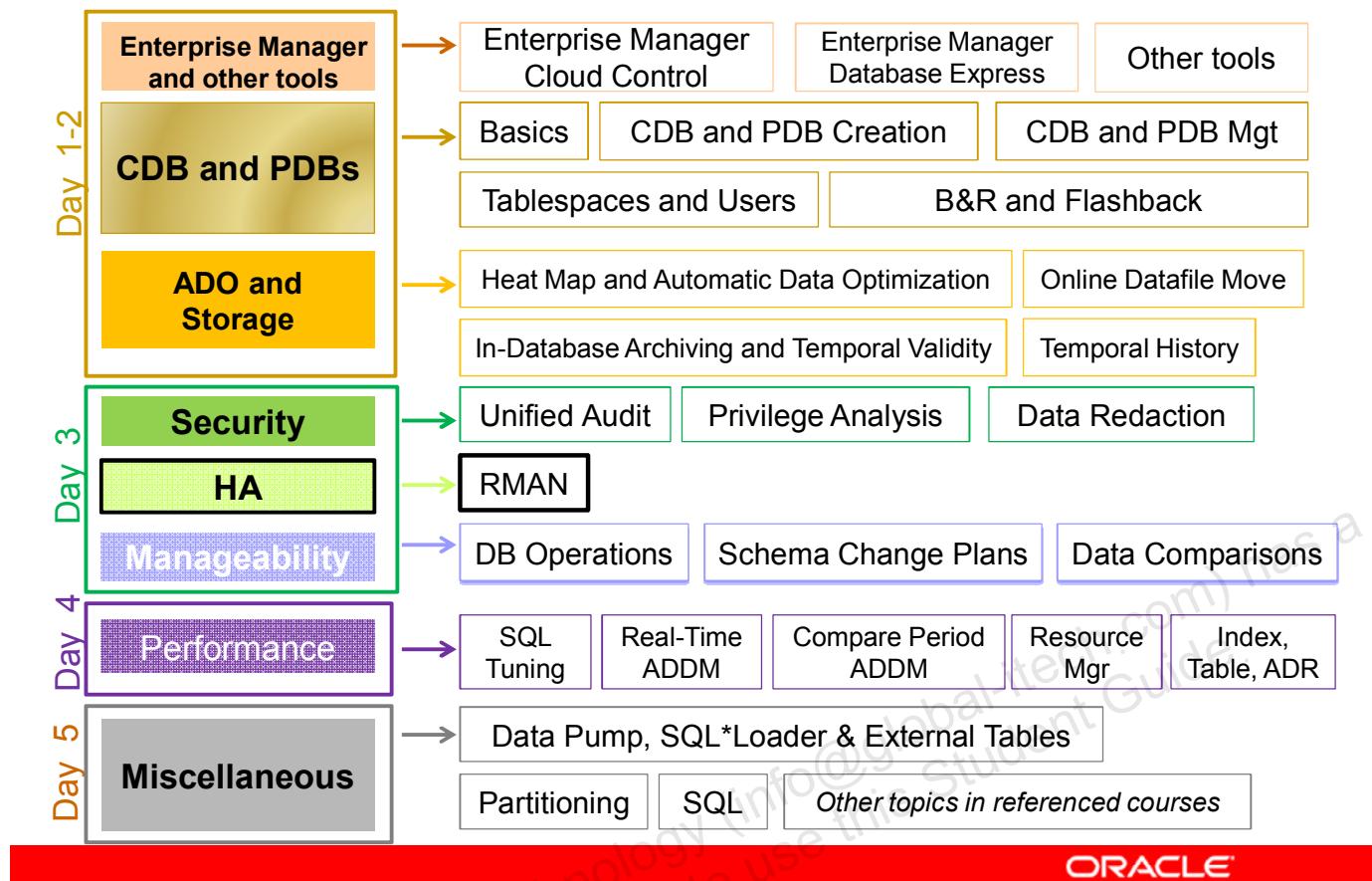
12

Recovery Manager New Features

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several enhancements for **Recovery Manager**.

Objectives

After completing this lesson, you should be able to:

- Describe the new RMAN enhancements
- Perform cross-platform data transport
- Use Table Recovery



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of the Oracle Recovery Manager and Flashback Data Archive enhanced features usage, refer to the following guide in the Oracle documentation:

- *Oracle Database Administration Guide 12c Release 1 (12.1) – “Transporting Data” chapter*
- *Oracle Database Backup and Recovery User’s Guide 12c Release 1 (12.1) – “Configuring Privilege and Role Authorization” chapter*
- *Oracle Database Backup and Recovery Reference 12c Release 1 (12.1)*
- *Oracle Database Security Guide 12c Release 1 (12.1) – BACKUP, RESTORE and RECOVER chapters*

Refer to other sources of information:

- *Oracle Database 12c New Features Demo Series* demonstrations under Oracle Learning Library:
 - *Transporting Data*
 - *Table Recovery*

Separation of DBA Duties

| RMAN Enhancements |
|-------------------|
| SYSBACKUP |
| RMAN SQL |
| RMAN DESCRIBE |
| Multisection |
| Duplication |
| Third-party |

Introducing task-specific and least-privileged administrative privilege **SYSBACKUP**

- Includes permissions for backup and recovery (connecting to a closed database)
- Does not include data access privileges such as SELECT ANY TABLE
- Is granted to the SYSBACKUP user that is created during database installation
- Can be explicitly used in RMAN connection by a SYSBACKUP privileged user



```
% rman TARGET '"john@orcl AS SYSBACKUP"'
connected to target database: ORCL (DBID=1297344416)
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c provides support for separation of database administration (DBA) duties for the Oracle database by introducing task-specific and least-privileged administrative privileges that do not require the SYSDBA administrative privilege. The new privilege to connect and execute commands in Recovery Manager is the SYSBACKUP privilege.

If you are working in a multitenant container database, each PDB of the CDB can create a local user and grant the user the SYSBACKUP privilege. The user can connect to the PDB and perform the backups for the PDB and only for the PDB he can connect to.

```
$ rman TARGET '"tim@pdb1 AS SYSBACKUP"'
```

In this case, to backup the whole database, use the BACKUP DATABASE command with no database name. You are not allowed to use the BACKUP PLUGGABLE DATABASE command.

Using SQL in RMAN

- No SQL prefix or quotes required
- Provides SQL*Plus DESCRIBE functionality
- Earlier releases:

| RMAN Enhancements |
|-------------------|
| SYSBACKUP |
| RMAN SQL |
| RMAN DESCRIBE |
| Multisection |
| Duplication |
| Third-party |

```
RMAN> sql 'ALTER TABLESPACE users
ADD DATAFILE ''/testdata/users02.dbf'' SIZE 10M';
```

- With Oracle Database 12c

```
RMAN> ALTER TABLESPACE users
      ADD DATAFILE '/testdata/users02.dbf' SIZE 10M;
```

```
RMAN> SELECT NAME, DBID, LOG_MODE FROM V$DATABASE;
```

| Name | Null? | Type |
|-----------|-------|----------------|
| TEST_NAME | | VARCHAR2 (128) |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

What you already know

You can execute SQL commands and PL/SQL procedures from the RMAN command line. In versions earlier than Oracle Database 12.1, this required the SQL prefix and quotes as you see in the first example of adding a data file.

What is new

You can now issue most SQL commands in RMAN without preceding the command with the SQL keyword. As you see in the second example, this is easier to write.

An example of a simple SELECT command:

```
RMAN> select sysdate from dual;
SYSDATE
-----
14-FEB-12
```

You can also use the abbreviated version DESC or the spelled-out DESCRIBE to list the column definitions of a table or view as shown in the slide.

Find the list of the SQL commands in *Oracle Database Backup and Recovery Reference 12c Release 1 Chapter RMAN Commands - SQL*

Backing Up and Restoring Very Large Files

| RMAN Enhancements |
|-------------------|
| SYSBACKUP |
| RMAN SQL |
| RMAN DESCRIBE |
| Multisection |
| Duplication |
| Third-party |

Multisection backups of a single data file:

- Are created by RMAN, with your specified size value
- Are for backup sets and **image copies**
- For full and **incremental backups**

Benefits:

- Reduce image copy creation time
- Are processed independently (serially or in parallel)
- Benefit Exadata

Requirements and restrictions:

- COMPATIBLE=12 . 0
- Not for control files or spfiles
- Not for a large value of parallelism



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

What you already know

Oracle data files can be up to 128 TB in size. By default, the smallest unit of an RMAN backup is an entire file. RMAN can optionally break up large files into sections and it can back up and restore these sections independently. You initiate the creation of multisection backups with the SECTION SIZE clause.

How this can help

Multisection image copies reduce the image copy creation time for large data files, in particular benefiting Exadata environments. This can also reduce completion time for non-backup use cases. For example, copying a file as part of transportable tablespace procedure or creating a clone with the active database duplication.

Each section can be processed independently, either serially or in parallel. Backing up a file into separate sections can improve the performance of the backup operation, and it also allows large file backups to be restarted.

Requirements and restrictions for multisection backups

- You can create multisection incremental backups only for data files, not for other database files, such as control files or spfiles.
- You should not apply large values of parallelism to back up a large file that resides on a small number of disks, as that would defeat the purpose of the parallel operation; multiple simultaneous accesses to the same disk device would be competing with each other.

RMAN Duplication Enhancements

| RMAN Enhancements |
|-------------------|
| SYSBACKUP |
| RMAN SQL |
| RMAN DESCRIBE |
| Multisection |
| Duplication |
| Third-party |

What is new for active database duplication:

- Default use of backup sets for active duplication
- Choice of compression, section size, and encryption
- Option to end database duplication with database in mounted, but not opened, state

Duplication of pluggable databases



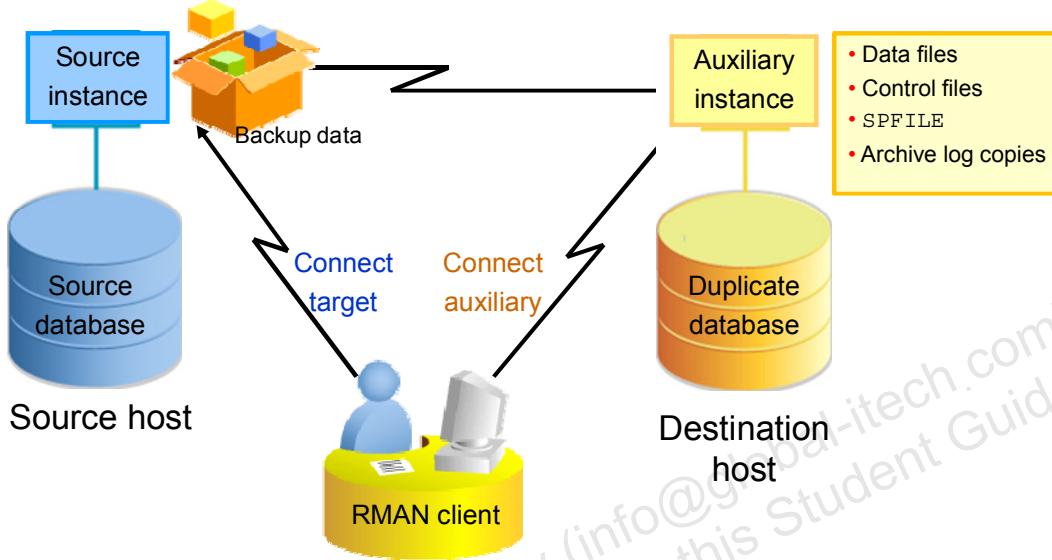
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The active database duplication process has additional options.

The RMAN DUPLICATE command allows to duplicate multitenant container databases and pluggable databases.

Duplicating an Active Database

- Via Enterprise Manager or RMAN command line
- With network (use the FROM ACTIVE DATABASE clause)
- With a customized SPFILE



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

What you already know about duplicating an active database

You can instruct the source database to send a “clone” of itself directly to the auxiliary instance by using Enterprise Manager or the FROM ACTIVE DATABASE clause of the RMAN DUPLICATE command.

Preexisting backups are neither needed, nor used for this operation. The online image copies are created by the source database and directly transmitted via Oracle Net (they are not written to disk) when using the FROM ACTIVE DATABASE clause of the RMAN DUPLICATE command. The source database can be open or mounted.

RMAN connects as TARGET to the source database instance and as AUXILIARY to the auxiliary instance (as shown in the slide).

The required files (data files, control files, SPFILE, and archive log copies) are copied from the source to an auxiliary instance via an inter-instance network connection. RMAN then uses a “memory script” (one that is contained only in memory) to complete recovery and open the database.

This method of active database duplication is referred to as the push-based method.

What Is New?

Default use of backup sets for active duplication

- Original « push » process based on image copies becomes « pull » process based on **backup sets**
- Automatic « pull » process when a clause is used:
 - Compression
 - Section size
 - Encryption

```
RMAN> SET ENCRYPTION ...;
```

Data is encrypted on the source database,
before transmission.

```
RMAN> DUPLICATE TARGET DATABASE TO orcl2
```

```
    FROM ACTIVE DATABASE
```

```
    [USING BACKUPSET]
```

```
    [SECTION SIZE ...]
```

```
    [USING COMPRESSED BACKUPSET] ...;
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The original “push” process is based on image copies.

With Oracle Database 12c, a “pull” (or restore) process is based on backup sets. A connection is first established with the source database. The auxiliary instance then retrieves the required database files from the source database as backup sets. A restore operation is performed from the auxiliary instance instance. Therefore, fewer resources are used on the source database. Both TNS connections are required on target and auxiliary instances.

Based on the `DUPLICATE` clauses, RMAN dynamically determines which process to use (push or pull). This ensures that existing customized scripts continue to function.

- When you specify `USING BACKUPSET`, RMAN uses the pull method.
- When you specify `SET ENCRYPTION` before the `DUPLICATE` command, RMAN automatically uses the pull method and creates backup sets. The backups sent to the destination are encrypted.
- The `SECTION SIZE` clause divides data files into subsections that are restored in parallel across multiple channels on the auxiliary database. For an effective use of parallelization, allocate more `AUXILIARY` channels.
- With the `USING COMPRESSED BACKUPSET` clause, the files are transferred as compressed backup sets. RMAN uses unused block compression while creating backups, thus reducing the size of backups that are transported over the network.

The NOOPEN Option

Option to end database duplication with database in mounted, but not opened, state:

- By default, opens the duplicate database with the RESETLOGS option, and creates the online redo log files for the duplicate database
- With NOOPEN option, the process finishes with the duplicate database in MOUNT mode

Potential NOOPEN use cases:

- Moving the location of the database (for example, to ASM)
- Upgrading a database (where the database must not be open with resetlogs, prior to running upgrade scripts)



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You might duplicate a database with RMAN for various reasons. In earlier versions a recovered duplicated database was automatically opened. By default, this functionality continues with the Oracle Database 12c.

What is new is that you have an option to finish the duplication process with the database in a mounted, but not opened state. This is useful when the attempt to open the database would produce errors and in all cases when you want to modify initialization settings, which are otherwise quite difficult to modify.

For example, you may want to move the location of the database to ASM. Also when you are performing an upgrade, where the database must not be open with resetlogs, prior to running upgrade scripts.

The NOOPEN option allows the duplication to create a new database as part of an upgrade procedure and leaves the database in a state ready for opening in upgrade mode and subsequent execution of upgrade scripts.

Duplicating Pluggable Databases

- A single pluggable database:

```
RMAN> DUPLICATE DATABASE TO cdb1 PLUGGABLE DATABASE pdb1;
```

- Several pluggable databases:

```
RMAN> DUPLICATE DATABASE TO cdb1
      PLUGGABLE DATABASE pdb1, pdb3;
```

- All pluggable databases except one:

```
RMAN> DUPLICATE DATABASE TO cdb1
      SKIP PLUGGABLE DATABASE pdb3;
```

- A PDB and tablespaces of other PDBs:

```
RMAN> DUPLICATE DATABASE TO cdb1
      PLUGGABLE DATABASE pdb1
      TABLESPACE pdb2:users;
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

What Is New?

RMAN enables you to duplicate pluggable databases by using the DUPLICATE command. When you duplicate a pluggable database, by default, RMAN duplicates the root and the seed database of the multitenant container database that contains the pluggable database.

The first example shows how to duplicate a single pluggable database, the second one how to duplicate a set of pluggable databases, the third one how to duplicate all the databases in the multitenant container database, except a pluggable database, and the last one how to duplicate a set of tablespaces within a pluggable database.

You must be logged in to the root as a user who is granted the SYSDBA or SYSBACKUP role.

Find the whole procedure in *Oracle Database Backup and Recovery User's Guide 12c Release 1 Chapter Duplicating a Database*

Recovering Databases with Third-Party Snapshots

| RMAN Enhancements |
|-------------------|
| SYSBACKUP |
| RMAN SQL |
| RMAN DESCRIBE |
| Multisection |
| Duplication |
| Third-party |

- Recover a database using the most recent snapshot:

```
RMAN> RECOVER DATABASE;
```

- Recover a database using a particular snapshot:

```
RMAN> RECOVER DATABASE UNTIL TIME '02/15/2012 11:00:00'
      SNAPSHOT TIME '02/15/2012 10:00:00';
```

- Perform a partial recovery using archived redo log files:

```
RMAN> RECOVER DATABASE UNTIL CANCEL
      SNAPSHOT TIME '02/15/2012 10:00:00';
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Third-Party Snapshots

You can use third-party technologies to take storage snapshots without putting the database or associated data files in BACKUP mode.

The snapshot technology stores the time at which the snapshot is completed.

Recover Using Snapshots

Use the RECOVER command using the new keyword SNAPSHOT TIME to recover a snapshot to a consistent point-in-time.

The SNAPSHOT TIME is used to inform the database that all of the files have been restored from a consistent snapshot.

- The first recovery example uses the most recent snapshot.
- The second example uses a specific snapshot, taken on February 15, 2012 at 10 AM. The UNTIL TIME clause can specify any time after the snapshot.
- The third example performs a partial recovery using archived redo log files from a snapshot taken on February 15, 2012 at 10 AM.

Note: You should periodically verify your backups to ensure that they are usable for recovery.

Quiz

In Oracle Database 12c, multisection backups can be used only for backup sets.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

Transporting Data Across Platforms

Transporting databases, data files, and tablespaces across platforms:

- In earlier releases, by using image copies
- Oracle Database 12c, **additionally by using backup sets**
- Allows **inconsistent tablespace** backups

Pre-12c

12c

Benefits:

- Reduced down time for platform migrations
- Choice of compression and multisection
- Not catalogued in control file, not used for regular restore operations



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

What You Already Know

RMAN enables you to transport databases, data files, and tablespaces across platforms. This includes transporting tablespaces across platforms with different endian formats. You can transport a whole database and thus convert a whole database on the destination host or source host as long as both the source and destination platforms share the same endian format. In earlier releases, cross-platform data transport uses only image copies, using RMAN CONVERT command.

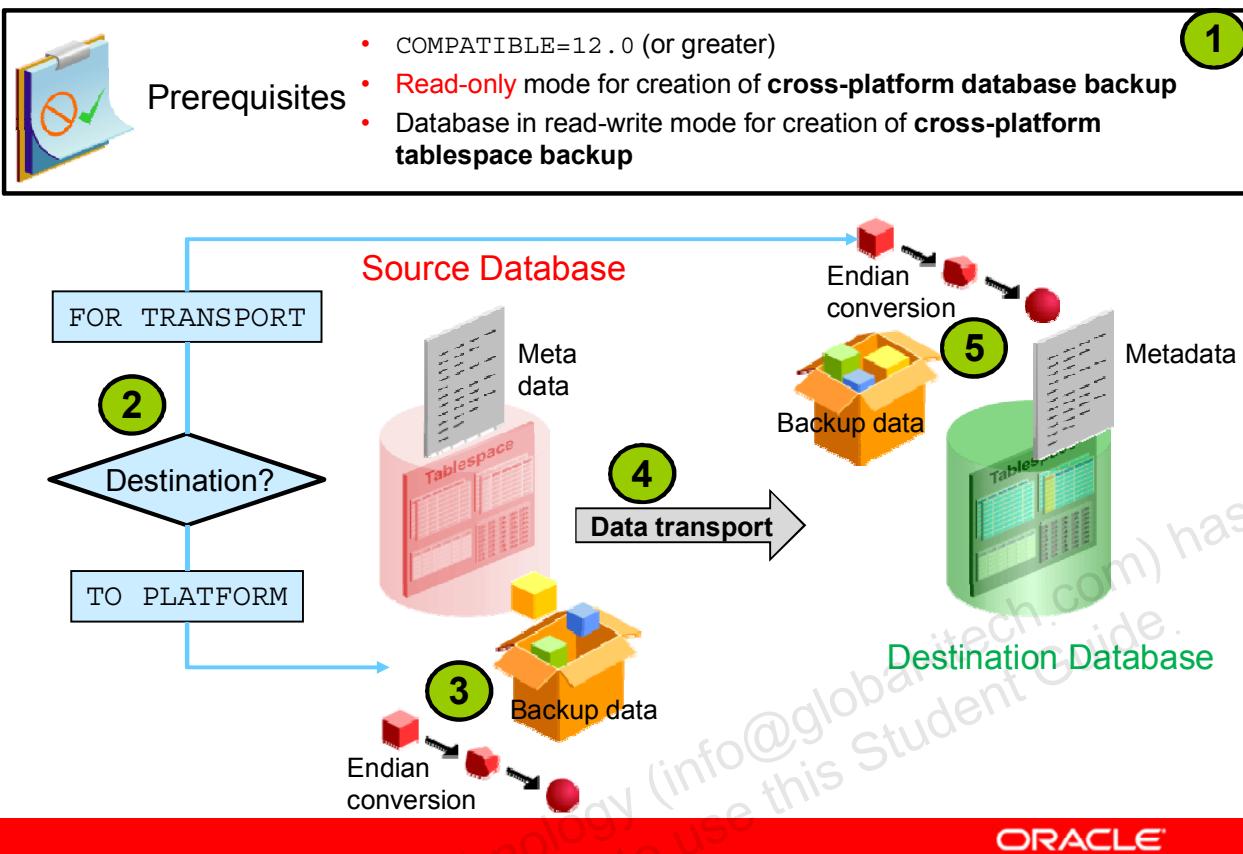
What Is New

Starting with the Oracle Database 12c, you can transport data across platforms by using backup sets. You can also create cross-platform inconsistent tablespace backups by using image copies and backup sets. An inconsistent tablespace backup is one that is created when the tablespace is not in read-only mode. This method is allowed with the new clause ALLOW INCONSISTENT.

With the use of backup sets, you can choose compression and multisection options, that reduce the overall transport time.

Note: RMAN does not catalog backup sets created for cross-platform transport in the control file. This ensures that backup sets created for cross-platform transportation are not used during regular restore operations.

Data Transport

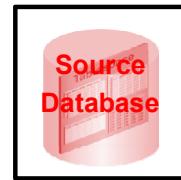


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. Before you create a cross-platform data transportation backup set, the following prerequisites must be met:
 - COMPATIBLE parameter must be set to 12.0 or greater.
 - To transport an entire database, the source database must be open in read-only mode, because the SYS and SYSAUX tablespaces participate in the transport.
 - To transport tablespaces only, use the DATAPUMP clause. The database must be open in read-write mode, so that Data Pump can access the metadata.
2. There are two alternatives that affect the location of the endian conversion. The FOR TRANSPORT clause creates a cross-platform backup indicating that the backup set can be transported to any destination database. The TO PLATFORM clause indicates that the conversion performs on the source database for a specific platform and can be restored on that specific platform. If you transport tablespaces only, the DATAPUMP clause is required to create an export dump file for the transported tablespaces during the last incremental backup of the tablespace in read-only mode.
3. Use the BACKUP FOR TRANSPORT or TO PLATFORM command to create a cross-platform data backup set on the source host.
4. Use any operating system-utilities to transfer the created backup set (and dump file).
5. Use the RESTORE FOREIGN command to restore the cross-platform data backup set on the destination host.

Transporting Database: Process Steps - 1



1. Verify the prerequisites:
 - COMPATIBLE: greater or equal 12.0
 - OPEN_MODE: READ ONLY
 2. Start an RMAN session to connect to the source database.
 3. Query the exact name of the destination platform from V\$TRANSPORTABLE_PLATFORM view.
 4. Back up the source database including control files:
 - Conversion on the source host or
- ```
RMAN> BACKUP TO PLATFORM 'Linux x86 64-bit'
 FORMAT '/bkp_dir/transp.bck' DATABASE;
```
- Conversion at the destination host
- ```
RMAN> BACKUP FOR TRANSPORT FORMAT '/bkp_dir/transp.bck'
      DATABASE;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

1. Verify the prerequisites (the source database must be open in read-only mode if the whole database is transported).
2. Start an RMAN session and connect to the target instance.
3. For performing cross-platform database transport, you may need the exact name of the destination platform to which you are transporting data.

```
SELECT PLATFORM_ID, PLATFORM_NAME, ENDIAN_FORMAT
  FROM V$TRANSPORTABLE_PLATFORM
 WHERE UPPER(PLATFORM_NAME) LIKE '%LINUX%';
```

4. Back up the source database using the BACKUP command with TO PLATFORM or FOR TRANSPORT. The FORMAT clause indicates the directory where the backup sets containing the data required for cross-platform database transportation are stored on the source host.

In the first example, the conversion will take place on the source host and the files stored in /bkp_dir directory are converted for Linux x86 64-bit platform.

In the second example, the conversion will take place on the destination host during the restore command and the files stored in /bkp_dir directory on the source host are not converted yet.

Transporting Database: Process Steps - 2

5. Disconnect from the source database.
6. Move the backup sets to the destination host.
7. Connect to the destination host as TARGET.
8. Restore the full backup set with the RESTORE command.



```
RMAN> RESTORE FOREIGN DATABASE TO NEW
      FROM BACKUPSET '/bkp_dir/transp.bck';
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

5. Disconnect from the source database.
6. Move the backup sets created by the BACKUP command to the destination host. You can use operating system utilities to move the backup sets from the source host to the destination host.
7. Connect to the destination host, to which the database must be transported, as TARGET. Ensure that the destination database is not mounted.
8. Use the RESTORE command to restore the data in the destination database.

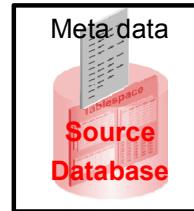
```
RESTORE FOREIGN DATABASE TO NEW
      FROM BACKUPSET '/bkp_dir/transp.bck';
```

Where:

- FOREIGN DATABASE indicates that the backup set is a cross-platform database backup.
- NEW specifies that the restored data files must use new OMF-specified names in the destination database. The backup set transp.bck can be found in the /bkp_dir directory of the destination host.

Transporting Tablespace: Process Steps - 1

1. Verify the prerequisites.
2. Start an RMAN session in the source database.
3. Query the exact name of the destination platform.
4. Make the tablespace **read-only**.



```
RMAN> ALTER TABLESPACE test READ ONLY;
```

5. Perform a cross-transportable backup and a Data Pump export.
 - Conversion on the destination host

```
RMAN> BACKUP FOR TRANSPORT FORMAT '/bkp/test.bck'
      DATAPUMP FORMAT '/bkp/test.dmp' TABLESPACE test;
```

- Conversion on the source host

```
RMAN> BACKUP TO PLATFORM 'HP Tru64 UNIX'
      FORMAT '/bkp/test.bck'
      DATAPUMP FORMAT '/bkp/test.dmp' TABLESPACE test;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

1. Verify the prerequisites: the source database must be opened in read-write mode.
2. Start an RMAN session and connect to the target instance.
3. For performing cross-platform tablespace transport, you may need the exact name of the destination platform to which you are transporting data.

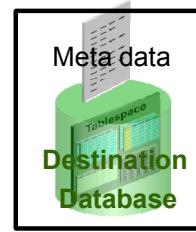
```
SELECT PLATFORM_ID, PLATFORM_NAME, ENDIAN_FORMAT
  FROM V$TRANSPORTABLE_PLATFORM
 WHERE UPPER(PLATFORM_NAME) LIKE '%LINUX%';
```

4. Make the tablespace read-only.
5. Back up the source tablespace using the BACKUP command with TO PLATFORM or FOR TRANSPORT to indicate where the conversion takes place. Use the DATAPUMP clause to indicate that an export dump file for the tablespaces must be created to export the tablespace metadata.

Note: A new BACKUP clause ALLOW INCONSISTENT enables you to back up tablespaces that are not in read-only mode. Although the backup is created, you cannot plug these tablespaces directly into the target database because they are inconsistent. You must later create an incremental backup of the tablespaces when they are in read-only mode. This incremental backup must contain the DATAPUMP clause that creates an export dump file of the tablespace metadata.

Transporting Tablespace: Process Steps - 2

6. Move the backup sets and the Data Pump export dump file to the destination host.
7. Connect to the destination host to the TARGET.
8. Restore the cross-transportable backup and the Data Pump export.



```
RMAN> RESTORE FOREIGN TABLESPACE test
      FORMAT '/oracle/test.dbf'
      FROM BACKUPSET '/bkp/test.bck'
      DUMP FILE FROM BACKUPSET '/bkp/test.dmp' ;
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

6. Disconnect from the source database and move the backup sets and the Data Pump export dump file to the destination host. You can use operating system utilities to move the backup sets and the dump file from the source host to the destination host.
7. Connect to the destination host, to which the tablespace must be transported, to the TARGET. Ensure that the destination database is opened in read-write mode.
8. Use the RESTORE command to restore the cross-transportable backup in the destination database as shown on the slide where FOREIGN TABLESPACE is the HP source data file. The FORMAT indicates where to restore the file retrieved from the backup set. Use the DUMP FILE FROM BACKUPSET to restore the metadata from the dump file required to plug the tablespace into the destination database.

Note: You can use a method that allows you to:

1. Create cross-platform inconsistent incremental backups with a new clause ALLOW INCONSISTENT


```
BACKUP INCREMENTAL FROM SCN=2720649 FOR TRANSPORT
      ALLOW INCONSISTENT FORMAT '/home/u_incl.bkp' TABLESPACE
      users;
```
2. Restore inconsistent cross-platform tablespace backup using RESTORE FOREIGN TABLESPACE command.
3. Recover restored data files copies with cross-platform incremental backups using RECOVER FOREIGN DATAFILECOPY command.

Quiz

When you transport data across platforms in Oracle Database 12c, you can use only image copies, not backup sets.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

Table Recovery

- **Table recovery with TSPITR, Flashback**
- **Table recovery from RMAN backups**
 - Without affecting the remaining database objects
 - Reduces time and disk space, compared to the earlier workflow like TSPITR:
 1. Recover tablespace in a separate disk location.
 2. Export desired table.
 3. Import into original database.

Pre-12c

12c



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Oracle database provides a number of different methods of how you can recover a table. It has done so in earlier versions as well. But the recovery scope might be much larger than what you want. The next interactive page provides an opportunity for you to review the major methods of:

- Database point-in-time recovery (**DBPITR**)
- Tablespace point-in-time recovery (**TSPITR**)
- Flashback Technology

What Is New

- With 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. This is a table recovery using RMAN backup data.
- This new functionality reduces time and disk space, compared to the earlier workflow:
 1. Recover tablespace in a separate disk location.
 2. Export desired table.
 3. Import into original database.

Recovering Tables from Backups

When to recover tables and table partitions from RMAN backups:

- It is recommended to NOT use TSPITR for the following situations:
 - Small number of tables
 - Not in a self-contained tablespace
- It is not possible to flash back under these conditions:
 - Purged tables (no Flashback Drop)
 - Beyond the available undo (no Flashback Table)
 - After a structural DDL change (no Flashback Table)



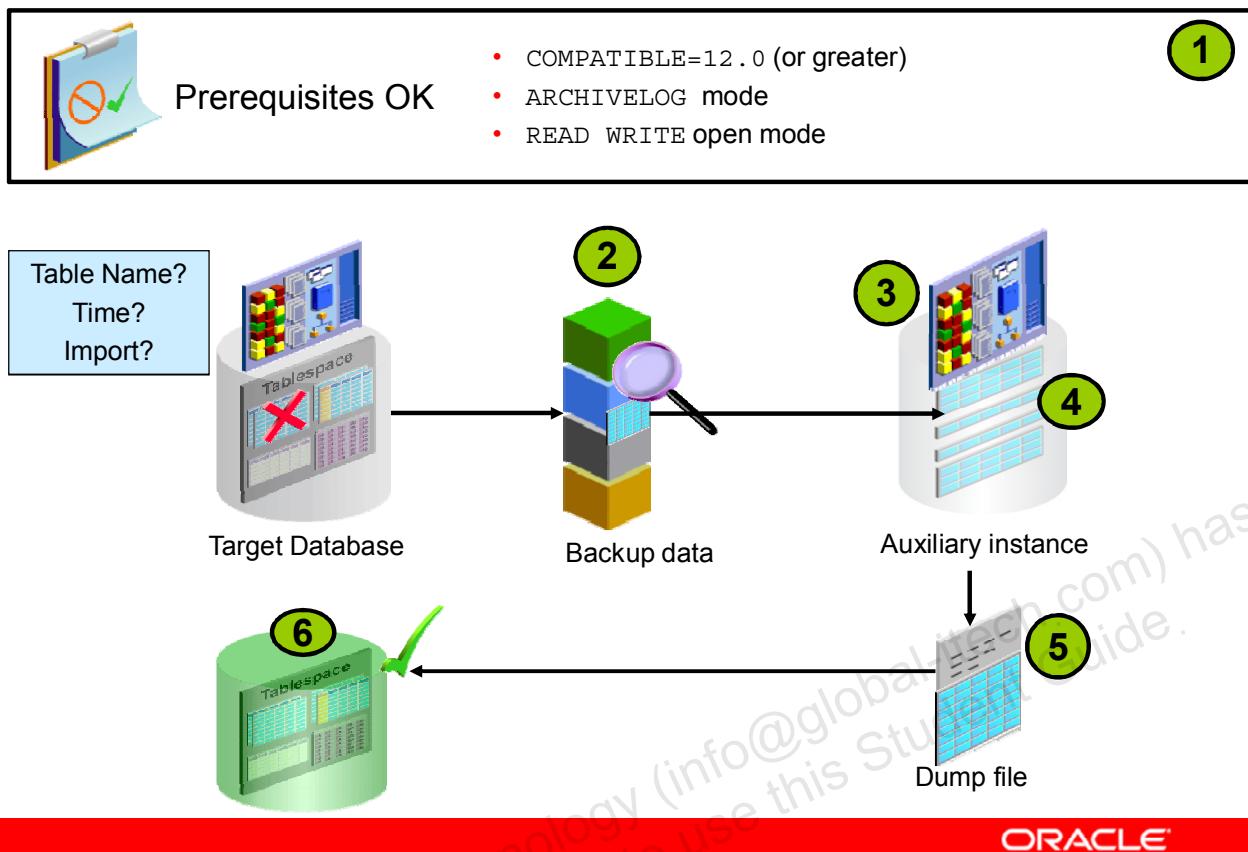
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

RMAN enables you to recover one or more tables or table partitions to a specified point in time without affecting the remaining database objects. Recovering tables and table partitions is useful in the following situations:

- You need to recover a very small number of tables to a particular point in time. In this situation, TSPITR is not the most effective solution because it moves all the objects in the tablespace to a specified point in time.
- You need to recover a tablespace that is not a self-contained to a particular point in time. TSPITR can be used only if the tablespace is self-contained.
- You need to recover tables that have either been corrupted or deleted with the PURGE option, so you cannot use the Flashback Drop functionality.
- You enabled logging for a Flashback Table, but the flashback target time or SCN is beyond the available undo.
- You want to recover data that is lost after a data definition language (DDL) operation has changed the structure of tables. You cannot use Flashback Table to rewind a table to before the point of a structural change, such as a truncate table operation.

Table Recovery: Graphical Overview



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

1. RMAN uses backups that were previously created to recover tables and table partitions to a specified point in time. The prerequisites are fulfilled and you provide the following input with the RECOVER command:
 - Names of tables or table partitions to be recovered
 - Point in time to which the tables or table partitions need to be recovered
 - Whether the recovered tables or table partitions must be imported into the target database
 RMAN uses your input to automate the process of recovering the specified tables or table partitions.
2. RMAN determines the backup based on your specification.
3. RMAN creates an auxiliary instance.
4. RMAN recovers your tables or table partitions, up to the specified point in time, into this auxiliary instance.
5. RMAN creates a Data Pump export dump file that contains the recovered objects.
6. RMAN imports the recovered objects into the target database.
7. You can customize this process as you will see in the following slides.

Specifying the Recovery Point-in-Time

Recover table and table partitions to the state they were in by specifying:

- UNTIL SCN *integer*: The system change number (SCN)
- UNTIL TIME '*date_string*': The time in the date format:
 - of the NLS_LANG and NLS_DATE_FORMAT environment variables, or:
 - Date constants, for example: SYSDATE - 5
- UNTIL SEQUENCE *integer* (THREAD *integer*): The log sequence number and thread number



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- RMAN recovers tables or table partitions to the state that they were at the time specified by the SCN. The SCN is an upper, non-inclusive limit.
- RMAN recovers tables or table partitions to the state they were in at the specified time. Use the date format specified in the NLS_LANG and NLS_DATE_FORMAT environment variables. You can also use data constants such as SYSDATE to specify the time. (SYSDATE - 5 means 5 days earlier than the system date).
- RMAN recovers tables or table partitions to the state they were at the time specified by the log sequence number and thread number. RMAN selects only files that it can use to restore or recover up to but not including the specified sequence number.

Process Steps of Table Recovery - 1

1. Perform the planning tasks and start an RMAN session with the CONNECT TARGET command.
2. Enter the RECOVER TABLE command.
3. RMAN determines the backup based on your specification.
4. RMAN creates an auxiliary instance by using the AUXILIARY DESTINATION clause, if specified.
5. RMAN recovers your tables or table partitions, up to the specified point in time, into this auxiliary instance.
6. RMAN creates a Data Pump export dump file that contains the recovered objects with the DUMP FILE=*name* and DATAPUMP DESTINATION=<OS path>.

Note: If a file with the name specified by DUMP FILE exists in the location in which the dump file must be created, then the export fails.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

1. After verifying the prerequisites, start an RMAN session and connect to the target instance.
2. Enter the RECOVER TABLE command with your required clauses.
3. RMAN determines the backup that contains the data that needs to be recovered, based on the point in time specified for recovery.
4. RMAN creates an auxiliary instance. Optionally, you can specify the location of the auxiliary instance files with the AUXILIARY DESTINATION or SET NEWNAME clauses of the RECOVER command. AUXILIARY DESTINATION is the recommended clause, because when you use SET NEWNAME and you would forget just one data file name, the recovery would not happen.
5. RMAN recovers the specified tables or table partitions, up to the specified point in time, into this auxiliary instance.
6. RMAN creates a Data Pump export dump file that contains the recovered objects. You can optionally specify the name of the export dump file (with the DUMP FILE clause, default OS-specific name) that is used to store the metadata from the source database. You can also specify the location in which the export dump file is created with the DATAPUMP DESTINATION clause. The location is typically the path of the OS directory that stores the dump file. If omitted, the dump file is stored in the AUXILIARY DESTINATION location. If that is not specified, then the dump file is stored in a default OS-specific location.

Customization

7. You can customize as follows:

- RMAN imports the recovered objects into the target database unless you specified NOTABLEIMPORT .
- RMAN optionally renames the recovered tables or table partitions with the REMAP TABLE and the REMAP TABLESPACE clauses.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

7. You can customize with the following options:

- By default, RMAN imports the recovered objects that are stored in the export dump file into the target database. However, you can choose not to import the recovered objects by using the NOTABLEIMPORT clause of the RESTORE command. If you choose not to import the recovered objects, RMAN recovers the tables or table partitions to the specified point and then creates the export dump file. However, this dump file is not imported into the target database. You must manually import this dump file into your target database, when required, by using the Data Pump Import utility.
- RMAN optionally renames the recovered tables or table partitions in the target database. When you recover tables or table partitions, you can either retain the existing names or rename the objects after they are imported into the target database. The REMAP TABLE option enables you to rename the recovered tables or table partitions in your target database. To import the recovered objects into a tablespace that is different from the one in which the objects originally existed, use the REMAP TABLESPACE clause of the RECOVER command. Only the tables or table partitions that are being recovered are remapped, the existing objects are not changed.

Quiz

When you perform table recovery from backup sets in Oracle Database 12c, RMAN automatically creates, uses, and deletes an auxiliary database instance.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe the new RMAN enhancements
- Perform cross-platform data transport
- Use Table Recovery



Practice 12 Overview: RMAN

These practices cover the following topics:

- SYSBACKUP administrative privilege and SQL and SQL*Plus commands in RMAN
- Table recovery

Module

Manageability



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

13

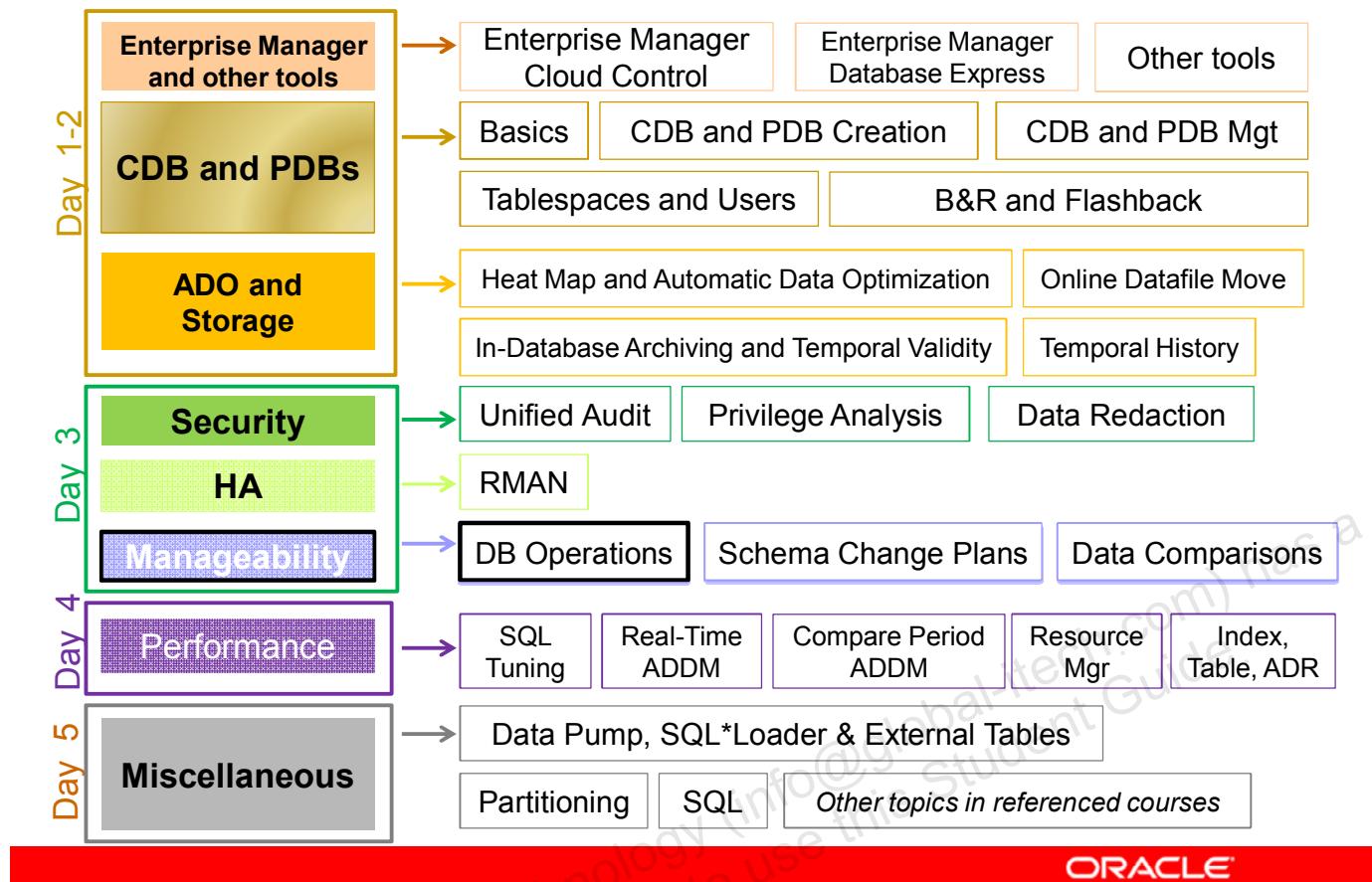
Real-Time Database Operation Monitoring



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Manageability: Managing databases implies monitoring database operations using the new Real-Time Database Operation Monitoring feature.

Objectives

After completing this lesson, you should be able to:

- Describe database operations
- Implement Real-Time Database Operation Monitoring



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of Real-Time Database Operation Monitoring new feature and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Performance Tuning Guide 12c Release 1 (12.1) – "Instance Tuning Using Performance Views – Real-Time Database Operation Monitoring Chapter"*
- *Oracle Database SQL Tuning 12c Release 1 (12.1) Chapter Monitoring Database Operations*

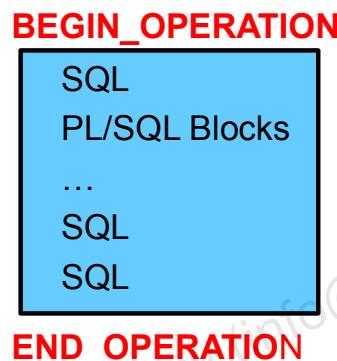
Overview

A database operation is:

- One or more SQL or PL/SQL statements
- In one or more sessions

A database operation can:

- Be monitored
- Produce active reports



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Real-Time Database Operation Monitoring extends and generalizes Real-Time SQL Monitoring.

Real-Time SQL Monitoring helps determine where a currently executing SQL statement is in its execution plan and where the statement is spending its time.

You can use Real-Time Database Operation Monitoring for performance monitoring of active SQL statements, PL/SQL procedures and functions. You can also see the breakdown of time and resources used for recently completed statements.

In databases where batch jobs run regularly, it can be very difficult to determine to which batch job an offending SQL statement belongs. In Decision Support Systems (DSS) it is not uncommon for many batch jobs to be running simultaneously. In some systems, hundreds of concurrent jobs could be running. Real-Time Database Operations Monitoring allows these jobs to be monitored while they are running.

Real-Time Database Operation Monitoring identifies an operation with a tag, determines what to monitor, and can generate reports. Real-Time Database Operation Monitoring monitors the progress of the operation, packages raw data for offline analysis and creates active reports that do not require access to production systems.

Use Cases

- Monitor batch jobs:
 - Set expected times ; alert when exceeded.
 - Identify and tune expensive statements.
- Diagnose background processes:
 - Identify and tune expensive statements.
- Diagnose changes to jobs over time:
 - Compare previous executions of an operation.
 - Compare changes to jobs after an upgrade.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In which situations would a DBA use Real-Time Database Operation Monitoring?

- A large retailer has a DSS with hundreds of batch jobs for Extraction, Transformation and Loading (ETL). You want to monitor these jobs in real time. As an Enterprise Manager user, you experience a very long wait on a page load. You want to easily identify what is running on behalf of the page load. At times a PL/SQL job that is required to finish in 24 hours is taking up to 80 hours to run. You need to identify and tune the expensive statements in the job. You want to be alerted if the job is expected to run too long, so it can be killed, fixed, and restarted before it uses too much time.
- After an upgrade a batch job went from 4.5 hours to 8 hours. You want to capture the performance statistics of the job steps, before and after the upgrade, then compare the two sets and identify the changes, for example, SQL execution plan, resource consumption, degree of parallelism.

Current Tools

- **Real-Time SQL Monitor**
 - Is a subset of Database Operation Monitoring
 - Monitors **long running single** SQL statements and PL/SQL functions
 - Displays statistics for recently completed statements
- **Active Session History**
 - 1-second sample
 - Useful for long running queries
- **SQL Trace**
 - Large overhead
 - Incident must be reproducible
- **DBMS_MONITOR** and **trcsess** utility
- **V\$SESSION_LONGOPS** and **DBMS_APPLICATION_INFO**



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- **Real-Time SQL Monitor**, a feature added in 11.1, is a subset of Database Operation Monitoring. Its main focus is to help DBAs know where a SQL statement is in its execution plan and where the time is being spent. It allows users to monitor the performance of SQL statements while they are executing as well as see the breakdown of time and resources used for recently completed statements. With SQL monitor, users can find the expensive (high response-time) SQLs (or PL/SQL functions) on their OLTP system, identify the expensive SQL statements in ETL, DDL, batch, reporting jobs, and check parallel-executed statements. It highlights poorly performed SQL: users can visualize hundreds of operations, find out why a particular operation is so expensive, and what is going on inside of a SQL execution.
- **Active session history (ASH)** contains samples of active sessions taken every second. It is on by default, so ASH can be used even if problems are not reproducible. Because samples are collected every second, ASH will contain more information for long running queries. Unless the queries are very long, ASH is not very useful for tuning response time.
- **SQL Trace** produces a lot of detailed information. The information is hard to interpret. The problem must be reproducible to enable SQL Trace to capture the details.
- **DBMS_MONITOR** package enables the trace for a given database session or module or action or service and **trcsess** command-line utility that consolidates tracing information.
- The **V\$SESSION_LONGOPS** view and **DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS** procedure are useful about long running operations to mainly indicate how much work has been processed so far.

Defining a DB Operation

Any set of database operations between two points in time identified by the user or application

Types of operations:

- Simple
 - One SQL or PL/SQL
- Composite
 - Multiple SQL statements and / or PL/SQL procedures and functions between two points in time
 - Single session
 - Multiple concurrent sessions



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Real-Time Database Operation Monitoring

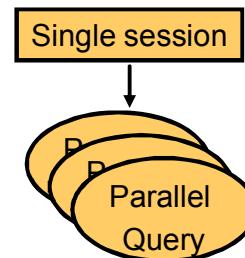
A DB operation is an operation that the database server runs to accomplish tasks. It can be either simple or composite:

- Simple database operations: Consist of one single SQL statement or one single PL/SQL function/procedure
- Composite database operations: Consist of the activity of one or more sessions between two points in time. SQL statements or PL/SQL procedures running in these sessions are part of the composite operations:
 - Single sessions: Single-session operations fall into two cases.
 - In one case, exactly one session exists for the duration of the database operation.
 - In the other case, multiple sessions exist in the database operation, but no more than one session runs at any given time. The application jumps from session to session.
 - Multiple concurrent sessions: In ETL, it is common for a job to involve multiple sessions running at the same time. You can define the entire ETL job as a single database operation.

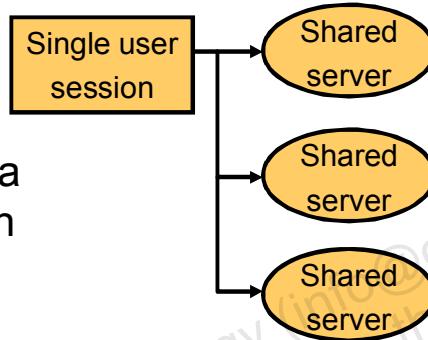
Scope of a Composite DB Operation

Single session

- Parallel query with single coordinator

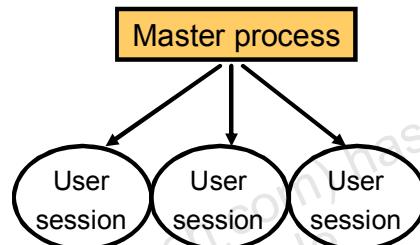


- Single-user session connected to a server session sequentially



Multiple sessions

- An operation using multiple user sessions concurrently



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A composite operation in the scope of a single session is composed of one of the following:

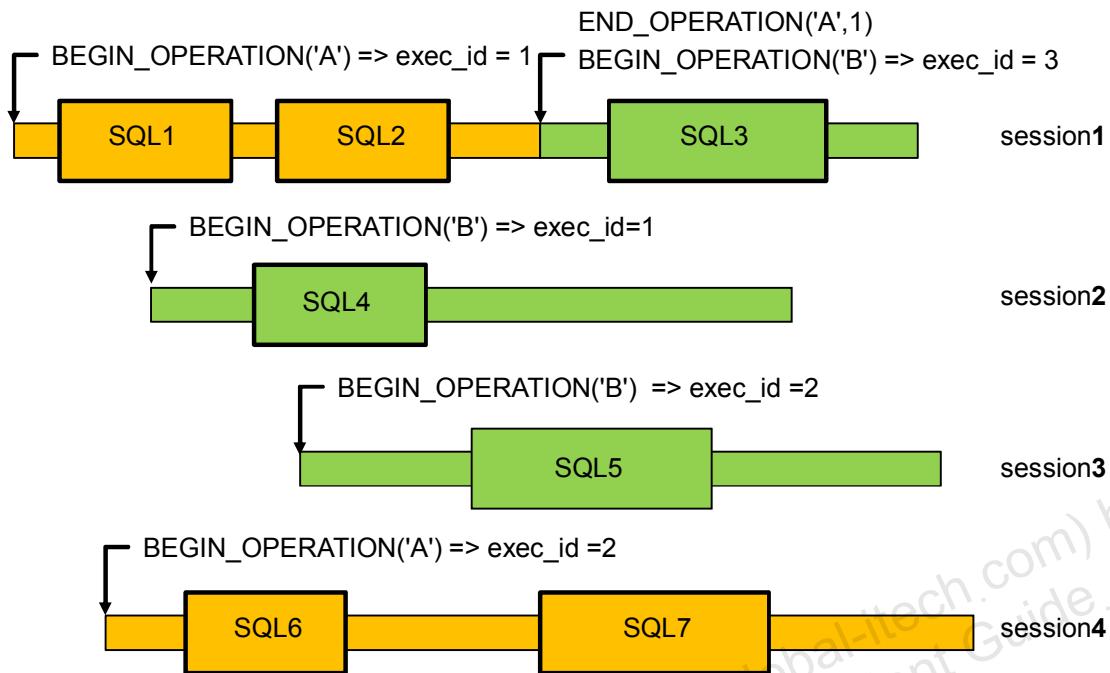
- SQL statements coordinated by the single session, such as parallel operations
- The active session, such as a user session in an application pool or shared server

Diagrams illustrating single-session and single-user session examples of a database operation are shown on the left side of the slide.

Note: These examples do not exclude a single session with statements that are neither non-parallel nor using shared servers.

A composite operation in the scope of multiple sessions is composed of SQL statements running concurrently such as during an ETL operation. This is shown in the diagram on the right side of the slide.

Database Operation Concepts



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In this illustration, there are four sessions. The first (upper) session shows that a session can belong to at most 1 database operation at a time. It belongs to database operation A at first, then belongs to database operation B. The `END_OPERATION` and then `BEGIN_OPERATION` commands causes the session to change database operations.

Sessions 1, 2, and 3 show that the name and execution ID are taken together to uniquely identify the database operation B.

The time that a session belongs to a database operation, but is not executing a SQL or PL/SQL statement is idle time.

Identifying a Database Operation

- Naming

DBOP (Name)



- Bracketing

DBMS_SQL_MONITOR.BEGIN_OPERATION



DBMS_SQL_MONITOR.END_OPERATION

ORACLE

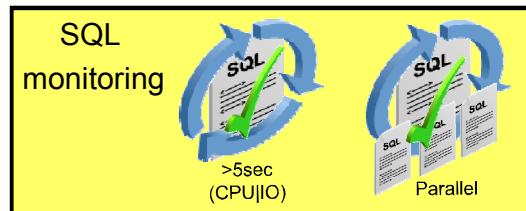
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To monitor a database operation, the database operation must be given a name, and a begin point and end point. There are two ways to name a database operation, the first is to insert a start point using `DBMS_SQL_MONITOR.BEGIN_OPERATION` function and an end point using `DBMS_SQL_MONITOR.END_OPERATION` procedure calls into the application. The second way is to set a tag, this is for JAVA and OCI applications. The tag can be set using the OCI calls `OCIAttrSet` and `OCIApplCtxSet`, the Java call `setClientInfo`, or from a OS environment variable `ORA_DBOP`. The tag in Java or OCI piggybacks the next database call so the begin point is easily distinguished.

The explicit calls provide a clear beginning and end of a database operation to provide the bracketing, but applications using tagging do not have an explicit method for ending an operation. Because starting a new operation ends the previous operation the tag should be set to `NULL` and sent to the database.

Enabling Monitoring of Database Operations

- At system level:
 - Set `STATISTICS_LEVEL` to `TYPICAL`
 - Set `CONTROL_MANAGEMENT_PACK_ACCESS` to `DIAGNOSTIC+TUNING`



- At database operation level:
 - Use `FORCED_TRACKING` attribute in `DBMS_SQL_MONITOR.BEGIN_OPERATION` function
- At statement level:
 - Use the `MONITOR` hint



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SQL monitoring feature is enabled by default when the `STATISTICS_LEVEL` initialization parameter is either set to `TYPICAL` (the default value) or `ALL`. Additionally, the `CONTROL_MANAGEMENT_PACK_ACCESS` parameter must be set to `DIAGNOSTIC+TUNING` (the default value) because SQL monitoring is a feature of the Oracle Database Tuning Pack. By default, SQL monitoring is started when a SQL command runs in parallel, or when it has consumed at least five seconds of the CPU or I/O time in a single execution. A SQL statement is also monitored when the `MONITOR` hint is explicitly declared in the statement.

When starting a database operation, you can force the database operation to be tracked when it starts. The default value of this parameter does not necessarily force any SQL statement or database operation to be tracked: the database operation is tracked only when it is sufficiently expensive. Use the `FORCED_TRACKING` attribute and set it to `Y` when starting the database operation with `DBMS_SQL_MONITOR.BEGIN_OPERATION` function:

```
DBMS_MONITOR.BEGIN_OPERATION (
    dbop_name IN VARCHAR2,
    dbop_eid IN NUMBER :=NULL,
    forced_tracking IN VARCHAR2 := NO_FORCE_TRACKING,
    attribute_list IN VARCHAR2 :=NULL)
RETURN NUMBER;
```

Identifying, Starting, and Completing a Database Operation

1. Identify the database operation.
 - Operation Name
 - Execution ID
2. Start the database operation with DBMS_SQL_MONITOR.BEGIN_OPERATION.
3. Complete the database operation with DBMS_SQL_MONITOR.END_OPERATION.

```
SQL> VAR dbop_eid NUMBER;
SQL> EXEC :dbop_eid := DBMS_SQL_MONITOR.BEGIN_OPERATION
          ('ORA.MV.refresh', FORCED_TRACKING => 'Y')
SQL> SELECT ...
SQL> SELECT ...
SQL> EXEC DBMS_SQL_MONITOR.END_OPERATION
          ('ORA.MV.refresh', :dbop_eid)
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When monitoring, each database operation can be identified by the operation name and execution ID. Every DB operation has a name and an execution ID. The DB operation name and execution ID together uniquely identify a specific execution of the operation. The operation name is specific to a piece of code. The same code may be executed by multiple sessions or users at the same time. Each invocation will have the same operation name but different execution IDs. The execution ID is not unique between operations, that is two DB operations with different names could have the same execution ID. The call to BEGIN_OPERATION returns the execution id. If an execution id is supplied to the call, it is returned, if the execution id is not supplied a unique execution ID is returned. In situations where a master process spawns several sessions that are associated with the same DB operation, an execution id would be generated for the entire operation and each session would set the execution id with a BEGIN_OPERATION call.

The database operation names are in a single namespace. A recommendation to prevent collisions is to use a format of <component>.<subcomponent>.<operation_name>. A suggestion is to use a component name of ORA for operations inside the database. The example on the slide is for an operation that performs a materialized view refresh.

The different executions of the same database operation (with the same name) can have different SQL statements or PL/SQL functions. This could be caused by different code paths being taken though conditions in PL/SQL. There is also nothing to prevent one operation name being assigned to multiple sets of code. This could cause confusion.

Monitoring the Progress of a Database Operation

The screenshot shows the Oracle Enterprise Manager Database Express interface. At the top, there's a navigation bar with links like Configuration, Storage, Security, and Performance. Below it is a performance hub report titled "Performance Hub: Real Time - Last Hour". The report includes a timeline chart showing resource usage over time and a table of "Monitored SQL" executions. One row in the table, labeled "Database Operation", is highlighted with a red box.

| Status | Duration | Type | SQL Plan Hash | User | Parallel | Database Time | IO Requests |
|--------|----------|---------------|---------------|------|----------|---------------|-------------|
| | 10.2m | ORA.HR.select | | | | 16.0ms | |
| | 5.9m | ORA.HR.select | | HR | | 7.6s | 81 |

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Enterprise Manager Cloud Control and Enterprise Manager Database Express allow you to view database operations that are active and recently finished.

Using Enterprise Manager Cloud Control, log in to the target database and click the Performance menu, then the SQL Monitoring option.

Using Enterprise Manager Database Express, in the Database Home page, the list of monitored SQL and database operations appears after drilling down into Monitored SQL menu.

In the Performance menu, click the Performance Hub option to get the list of monitored SQL and database operations, as shown in the second screenshot of the slide. For details, click the database operation name. On the Details page the execution ID is shown. By selecting the Database Operations filter above the list (SQL, PL/SQL and Database Operations), only database operations can be shown. Further filtering can be done on the operation name using the search box on the right above the list.

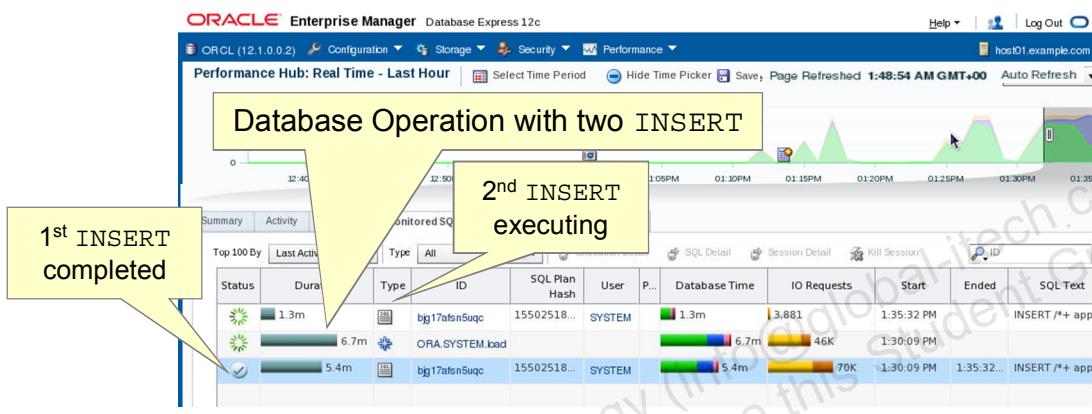
Monitoring Load Database Operations

```

SQL> VAR dbop_eid NUMBER;
SQL> EXEC :op_eid := DBMS_SQL_MONITOR.BEGIN_OPERATION
      ('ORA.SYSTEM.load', FORCED_TRACKING => 'Y')
SQL> INSERT /*+ APPEND */ * INTO tab1 SELECT ...;
SQL> INSERT /*+ APPEND */ * INTO tab2 SELECT ...;
SQL> EXEC DBMS_SQL_MONITOR.END_OPERATION
      ('ORA.SYSTEM.load', :op_eid)

```

One load is completed and the DB operation is still executing.



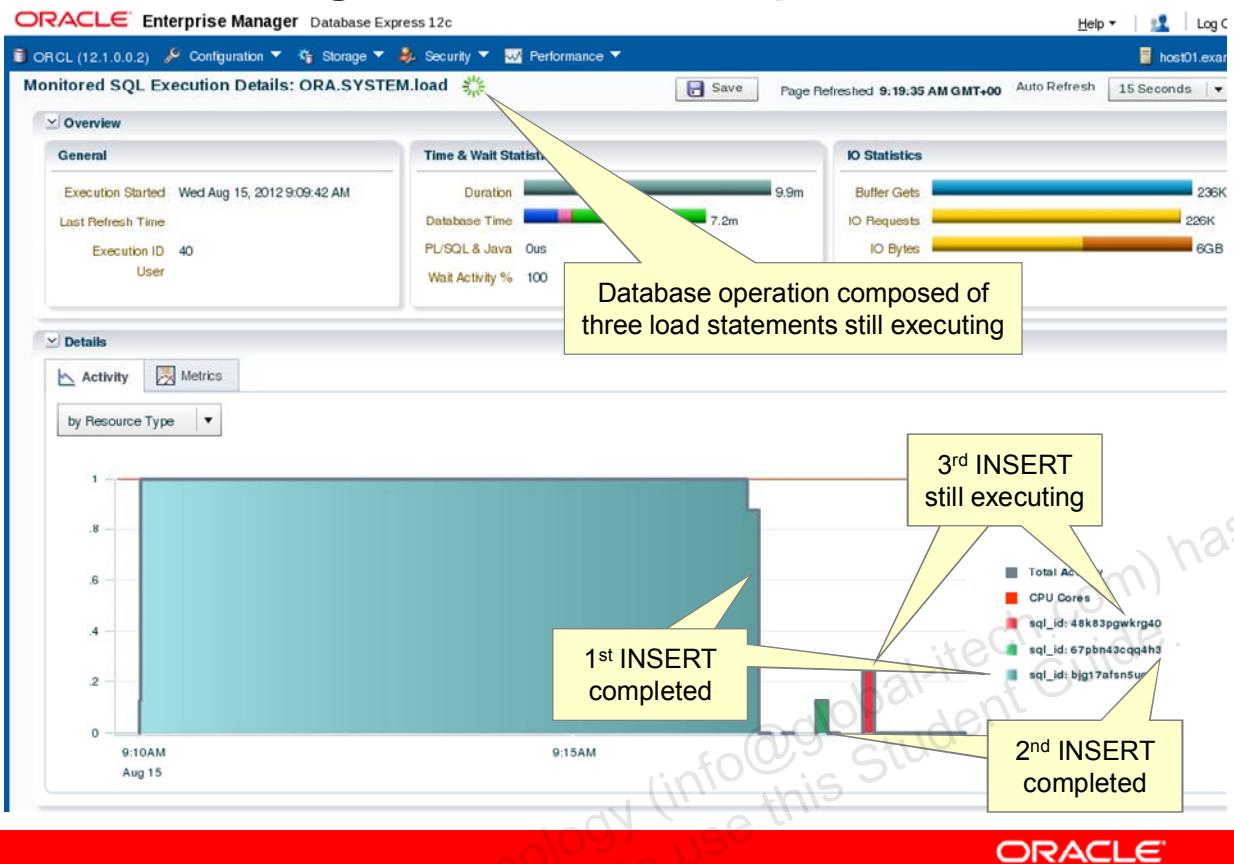
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Monitoring load operations can be performed. From the example shown on the slide, the database operation consists of two bulk load statements.

Using Enterprise Manager Database Express, from the Performance Hub page, you see that the database load operation is still executing whereas the first bulk load statement is already completed. Another bulk load statement is still executing, being part of the same database load operation being monitored.

Using Enterprise Manager Cloud Control, logged in to a target database, from the Performance menu and SQL Monitoring option, you would see the same database load operation executing.

Monitoring Load Database Operation Details



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

Reporting Database Operations Using Views

V\$SQL_MONITOR view shows the list of executing and completed database operations through new columns.

```
SQL> SELECT DBOP_NAME, DBOP_EXEC_ID, STATUS  
2   FROM V$SQL_MONITOR  
3  WHERE DBOP_NAME IS NOT NULL;
```

| DBOP_NAME | DBOP_EXEC_ID | STATUS |
|----------------|--------------|-----------|
| ORA.HR.select | 8 | EXECUTING |
| ORA.SH.select2 | 10 | EXECUTING |
| ORA.SH.select | 3 | DONE |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The V\$SQL_MONITOR view reports the list of executing and completed database operations including global, high-level information about the top SQL statements in a database operation. Each monitored SQL statement has an entry in this view. Each row contains a SQL statement whose statistics are accumulated from multiple sessions and all of its executions in the operation.

The primary key is the combination of the columns DBOP_NAME, DBOP_EXEC_ID, and SQL_ID.

Reporting Database Operations Using Views

- EM Cloud Control and EM Database Express use multiple views:
 - V\$SQL_MONITOR, V\$ACTIVE_SESSION_HISTORY and new columns: DBOP_NAME, DBOP_EXEC_ID
- AWR automatically captures SQL monitoring reports and stores active session history:
 - DBA_HIST_REPORTS, DBA_HIST_REPORTS_DETAILS
 - DBA_HIST_ACTIVE_SESS_HISTORY

```
SQL> SELECT session_id, DBOP_NAME, DBOP_EXEC_ID
  2  FROM DBA_HIST_ACTIVE_SESS_HISTORY
  3  where DBOP_NAME is not NULL;

SESSION_ID DBOP_NAME          DBOP_EXEC_ID
----- -----
  30 ORA.SYSTEM.load           8
  45 ORA.HR.select            11
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

EM Cloud Control or EM Database Express summarize the activity for monitored statements and database operations collected from multiple views.

- Recently completed and active database operations from:
 - V\$SQL_MONITOR, V\$ACTIVE_SESSION_HISTORY through new columns: DBOP_NAME and DBOP_EXEC_ID
 - V\$SQL_PLAN_MONITOR, V\$SQL_MONITOR_SESSTAT, V\$SQL, V\$SQL_PLAN, V\$SESSION_LONGOPS
- Completed database operations and reports are stored in AWR and can be reported in the following views:
 - DBA_HIST_REPORTS, DBA_HIST_REPORTS_DETAILS
 - DBA_HIST_ACTIVE_SESS_HISTORY

Reporting Database Operations Using Functions

DBMS_SQL_MONITOR package:

- REPORT_SQL_MONITOR_LIST_XML and REPORT_SQL_MONITOR_XML functions report the list and details of database operations in XML format.
- REPORT_SQL_MONITOR_LIST and REPORT_SQL_MONITOR functions report the list and details of database operations in a CLOB.

```
SQL> SELECT DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST_XML()
  2  FROM dual;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Reporting Using DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_xxx Functions

```
SQL> select DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST_XML() from
dual;

DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST_XML()
-----
<report db_version="12.1.0.0.2" cpu_cores="1" hyperthread="N"
timezone_offset="0
" elapsed_time="0.00" cpu_time="0.00">
<report_id><! [CDATA [/orarep/sqlmonitor/list?monitor_type=3] ]></rep
ort_id>
<sql_monitor_list_report version="1" sysdate="08/14/2012
10:33:04">
<dbop detail="F" dbop_name="ORA.SH.select2"
dbop_exec_start="08/14/2012 10:26:11" dbop_exec_id="12">
<status>EXECUTING</status>
<first_refresh_time>08/14/2012 10:26:11</first_refresh_time>
```

```
<refresh_count>0</refresh_count>
  <inst_id>1</inst_id>
<session_id>43</session_id>
  <session_serial>229</session_serial>
  <user_id>0</user_id>
  <con_id>0</con_id>
    ->
      <module>SQL*Plus</module>
      <service>SYS$USERS</service>
      <program>sqlplus@host01.example.com (TNS V1-V3)</program>
      <plsql_entry_object_id>10531</plsql_entry_object_id>
      <plsql_entry_subprogram_id>6</plsql_entry_subprogram_id>
      <plsql_object_id>10375</plsql_object_id>
      <plsql_subprogram_id>63</plsql_subprogram_id>
      <is_cross_instance>N</is_cross_instance>
      <stats type="monitor">
        <stat name="duration">413</stat>
        <stat name="elapsed_time">3072887</stat>
        <stat name="cpu_time">1143825</stat>
        <stat name="user_io_wait_time">368882</stat>
        <stat name="application_wait_time">475</stat>
        <stat name="other_wait_time">1559705</stat>
        <stat name="buffer_gets">23</stat>
        <stat name="read_reqs">37</stat>
        <stat name="read_bytes">13033472</stat>
        <stat name="write_reqs">52</stat>
        <stat name="write_bytes">42041344</stat>
      </stats>
    </dbop>
```

Database Operation Tuning

- ADDM
- SQL Tuning Advisor
- SQL Performance Analyzer



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The goal of tuning is to improve response time of a specific execution. Tuning database operation is more than just tuning SQL statements in the operation. For the first release, you will consider that database operation tuning is to tune the Top SQL statements one by one.

Here the tools you can use for SQL Tuning:

- The Automatic Database Diagnostic Monitor (ADDM): ADDM analyzes the information collected by the AWR for possible performance problems with Oracle Database, including high-load SQL statements.
- SQL Tuning Advisor and SQL Performance Analyzer: These tools can take SQL Tuning Sets or Top SQL statements as targets.

To run SQL Tuning Advisor or SQL Performance Analyzer on SQL statements from a PDB, a common user must have the following privileges:

- Common SET CONTAINER privilege or local SET CONTAINER privilege in the PDB
- The privileges required to execute the SQL statements in the PDB

Quiz

Monitoring database operations is allowed for sets of SELECT statements only.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

You can define any set of SQL and PL/SQL statements (including multiple, concurrent sessions) between two points in time as a database operation to be monitored.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe database operations
- Implement Real-Time Database Operation Monitoring

Practice 13 Overview: Monitoring Database Operations

These practices cover the following topics:

- Monitoring several database operations, including several SELECT statements, using Enterprise Manager Database Express or Enterprise Manager Cloud Control
- Monitoring a load database operation including several bulk load statements (*optional*)



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

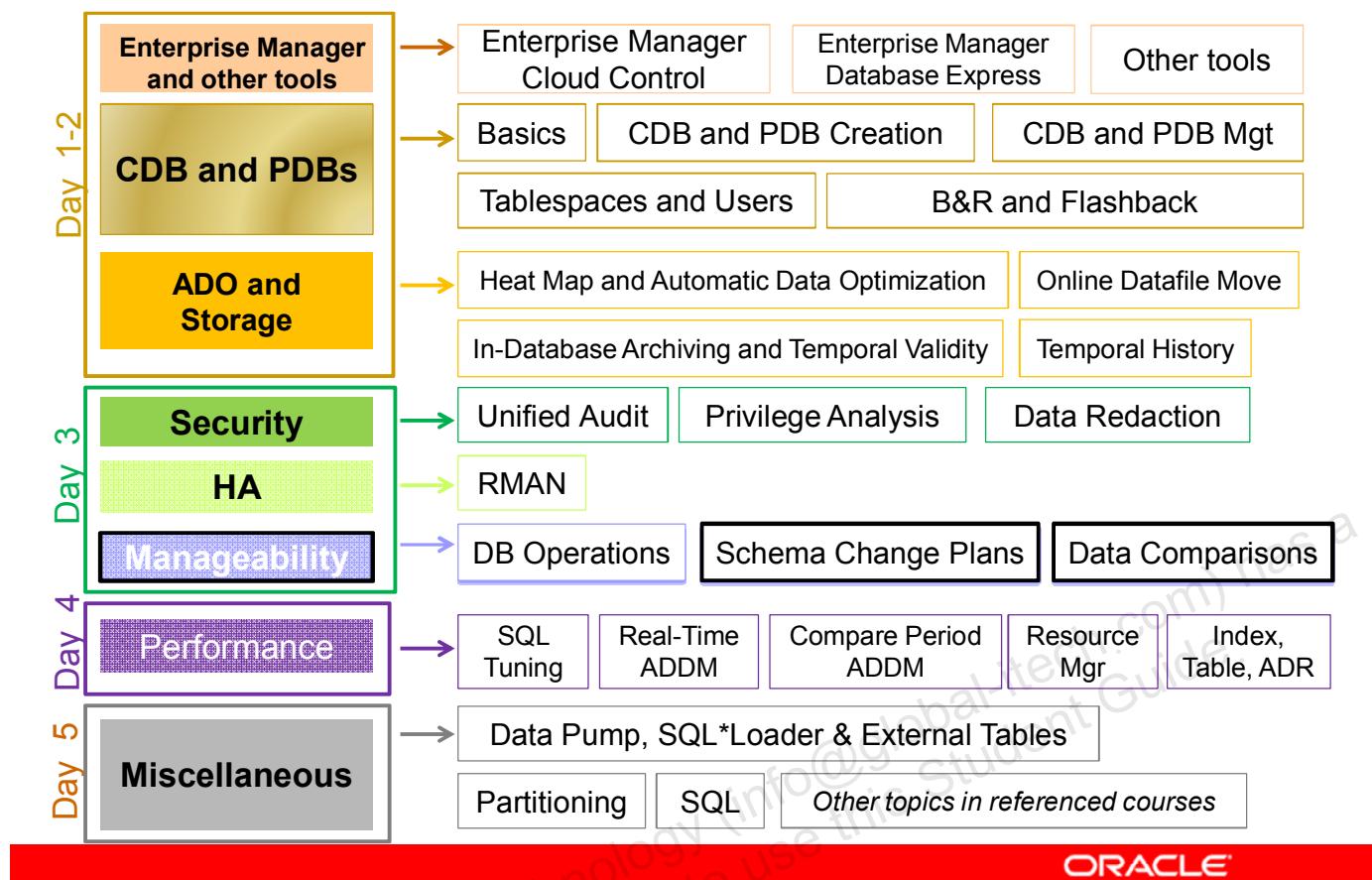
Schema and Data Change Management

14

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The **Oracle Enterprise Manager 11g Release 1 Change Management Pack** enables administrators to identify and implement planned database schema changes to support new application requirements without errors and data loss, as well as track unplanned schema changes that may be causing down time.

The **Oracle Enterprise Manager 12c Release 1 Database Lifecycle Management Pack** is a comprehensive solution that helps database, system, and application administrators automate the processes required to manage the Oracle Database Lifecycle. It eliminates manual and time-consuming tasks related to discovery, initial provisioning, patching, configuration management and ongoing change management.

The **Database Lifecycle Management Pack** includes the following features:

- **Schema and Data Comparisons:** Automate the comparison of database schema and data across Databases or saved baselines.
- **Schema and Data Synchronization:** Propagate database objects and schemas with or without data, and update database object definitions.
- **Schema Change Plans:** Allows developers to add changes to the schema in a plan that can be applied to multiple databases.
- **Schema Comparisons:** Compare database objects in two databases or baselines in Enterprise Manager or SQL developer.

Objectives

After completing this lesson, you should be able to explain:

- Why and when to use the new Schema Change Plans feature
- How to use schema change plans
- Why and when to use the new Data Comparisons feature
- How to use data comparisons



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For information about licensing the *Database Lifecycle Management Pack*, refer to the following guide in the Oracle documentation:

- *Oracle Enterprise Manager Licensing Information 12c Release 1 (12.1)*

Refer to other sources of information:

- *Oracle Enterprise Manager Cloud Control 12c Demo Series* demonstrations under Oracle Learning Library:
 - *Managing Schema Change Plans*
 - *Performing Data Comparison*

Schema Change Plans and *Data Comparison* operations are available only with Enterprise Manager Cloud Control.

Database Lifecycle Management Pack

New Features

Schema Change Plans:

- Save schema changes into a change plan.
- Apply change to multiple targets.



Data Comparisons:

- Compare reference data across environments.
- Identify differences between test and production.

The Oracle logo, which consists of the word "ORACLE" in a bold, white, sans-serif font, all-caps, with a registered trademark symbol (®) to the right.

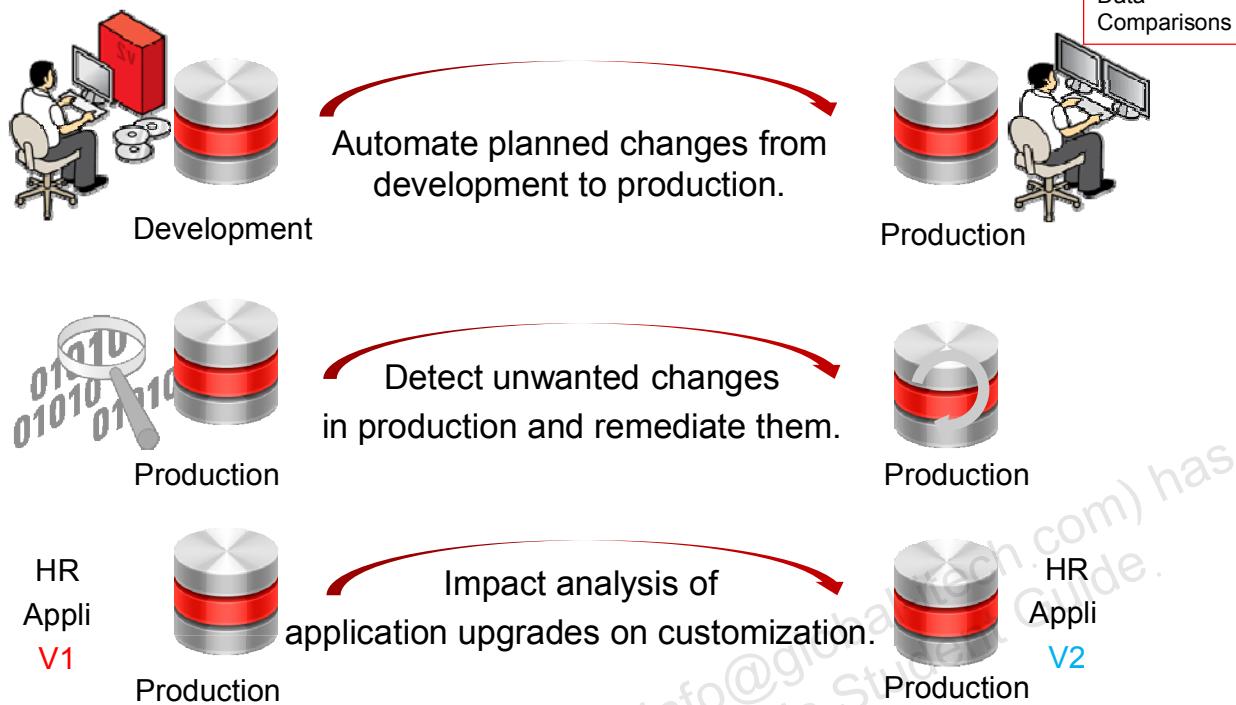
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Database Lifecycle Management Pack is all about making sure that enterprises can manage all changes, either proactively as they promote a database from development test to production, or reactively when any unwanted changes are made in a production environment that are causing problems. Enterprise Manager provides this capability to automatically identify and detect these changes so that organizations can take immediate actions and automate the process of applying the corrective actions without scripts and manual intervention.

The Database Lifecycle Management Pack allows you to:

- Group schema changes together into what is called a schema change plan based on comparison between two databases and propagate those change plans from one environment to another
- Perform data comparisons where you can compare data across environments, and identify the divergences in data between your test and production environments

Change Management Pack Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Change Management Pack Capabilities

The Oracle Enterprise Manager 11g Release 1 Change Management pack gives database administrators the ability to evaluate, plan for, and implement database schema changes to support new application requirements without error and data loss while minimizing down time.

The Oracle Enterprise Manager 11g Release 1 Change Management pack allows you to:

- Automate plan changes from development into production
- Detect unwanted changes in production and remediate them
- Analyze the impact of an application upgrade on customization

For example, when you have a custom application or a custom object or module and perform an application upgrade, there are some triggers added behind the scene, and columns may be added to the tables. Some modules may also be affected by database objects changes. Changes on your application objects are analyzed and reported.

How do you automate planned changes from development to production?

Change Management Pack Components

1. Create Dictionary Baselines:
 - Contain the definitions of a set of objects captured at a specific time
 - Can be used for comparisons
2. Perform Dictionary Comparisons:
 - Detect discrepancies between two versions of objects definitions
3. Perform Dictionary Synchronizations:
 - Make the definitions of a set of objects in the destination database the same as those in the source



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Enterprise Manager 11g Release 1, the key features of the Change Management pack that enable administrators to evaluate, plan, and implement database schema changes are the following:

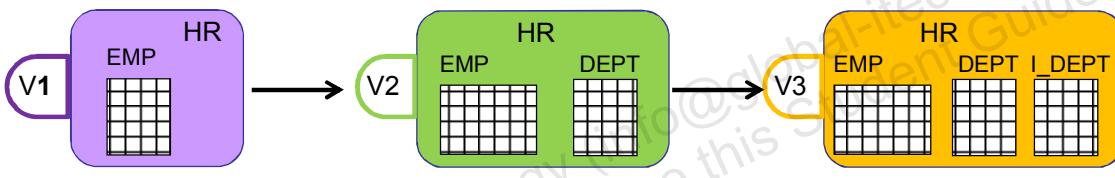
- Dictionary baselines: A point in time of the definition of the database and its associated database objects
- Dictionary comparisons: A complete list of differences between a baseline or a database and another baseline or a database
- Dictionary synchronizations: A process of promoting changes from a database definition capture in a baseline or from a database to a target database

Dictionary Baselines

Contain definitions of a database or subset of a database captured at a specific time

- Non-schema object types:
 - Tablespaces
 - Users, profiles, roles, grants
- All schema objects for an application:
 - Tables, views, indexes
 - Procedures, packages, triggers
 - init.ora

Provides point-in-time snapshots stored in the EM repository



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A dictionary baseline is an object that contains a set of database definitions captured at a certain point in time and stored in the Enterprise Manager repository.

When creating a baseline, you create a corresponding baseline scope specification, which describes the names and the types of database objects and schemas from which they should be captured.

At a later time, or at regular intervals, you capture additional versions of the same baseline. A scope specification identifies the database objects to be captured in a baseline. (Scope specifications also identify objects to process in dictionary comparisons and synchronizations.) When the scope of a baseline specified, you cannot change the scope specification. This restriction ensures that all versions of the baseline are captured using the same set of rules, which means that differences between versions result from changes in the database, not scope specification changes. The baselines can be used as references for future comparisons with other databases or other baselines to analyze if changes took place.

- Baseline scope specification can include:
 - Schemas and object types to capture
For example, you can capture all tables, indexes and views of schemas HR and SH
 - Non-schema objects such as users, roles and tablespaces, and privilege grants to users and roles
You can specify schemas to exclude, and object types.
 - Individual schema objects by specifying the type, schema and name of each object
 - init.ora parameters

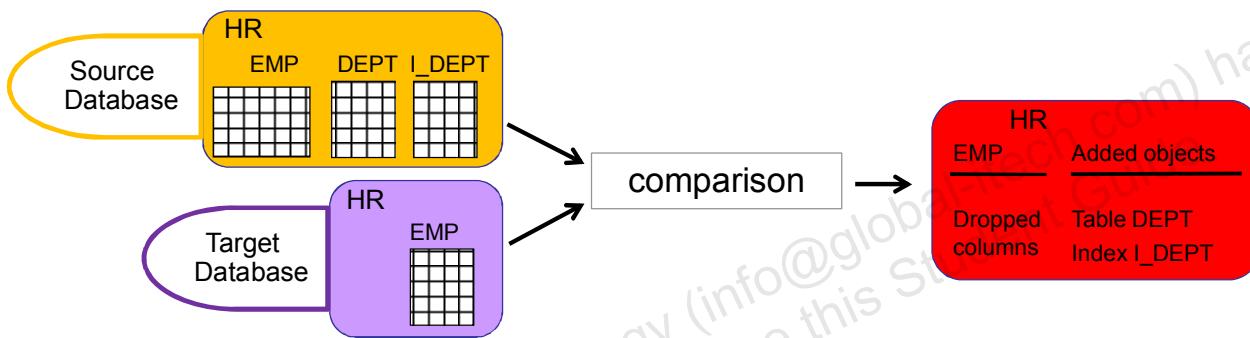
Baselines are good references to compare to other baselines or databases.

How Would Developers Use Baselines?

Several developers work on the same test database and make some changes in the same application. At different points in time, an automated job regularly creates a new version of the baseline, including changes made by the different developers over time. The reported changes are stored outside the database, in the Enterprise Manager repository. It is auditable.

Dictionary Comparisons

- Contain the results of comparing the definitions of database objects at a specific time
- Can compare:
 - Database with database
 - Baseline with database
 - Schema with schema
- Can define what you want to compare



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The next step in the process of propagating the identified changes to another database is to compare between the reference database or baseline and the target database.

You will perform a dictionary comparison that identifies differences in database object definitions between a baseline from the development database and the production database, or the test database and the production database, or two schemas within a single database/baseline.

A comparison specification is defined by a source and a target, scope, and owner. The scope specification describes the names and types of database object definitions to be included in the comparison and the schemas that contain these object definitions.

Comparisons identify differences in any attribute value between objects of any type. For example, comparisons show the differences between the definitions in the original baseline for the application and those in the current database. After creating a new comparison version, it identifies the differences between the original definitions at the start of the development cycle, and those same definitions at the current time.

Schema objects in the source are compared to objects in the same schema in the target. For example, the table `HR.EMP` in the development source database is compared to `HR.EMP` in the production target database.

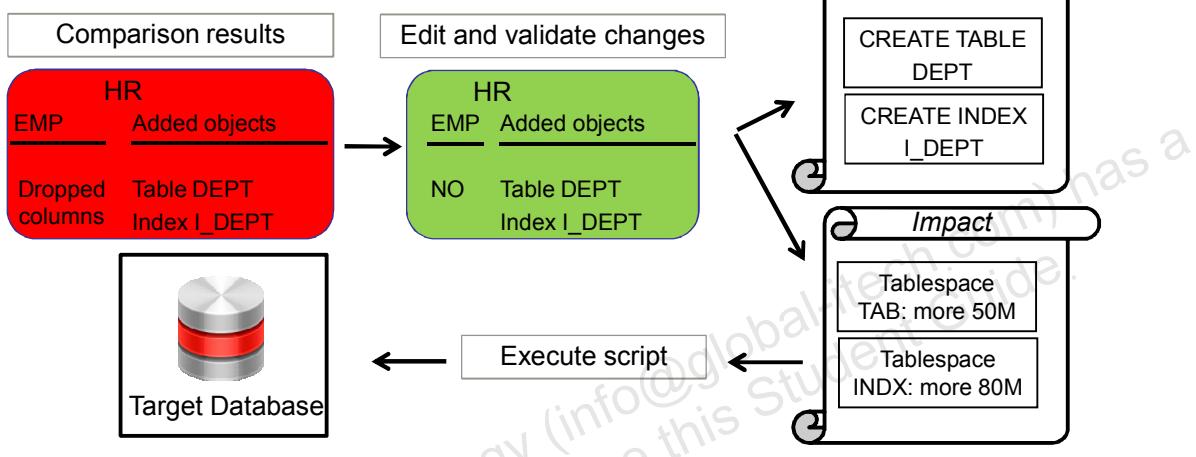
Any discrepancy between the source and target databases is reported:

- A new column is added to or dropped from a table in either one of the source databases or target database.
- A new index is created in the source database and not in the target database or vice versa.

Dictionary Synchronization

Dictionary synchronization includes three stages after compare source database or baseline with target:

1. View and edit validated changes before applying.
2. Generate impact report and script.
3. Execute of the script.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The last step is the dictionary synchronization that will synchronize differences in database object definitions between two databases, or a baseline and a database. The basic action of a database synchronization is to create or modify selected object definitions in a database to match those in another database or in a baseline.

Synchronization is performed in three steps:

1. View the list of suggested changes to selectively exclude objects from synchronization, before the generation of the script. Interactive synchronization mode gives you this chance at the second step to examine the results of comparison.
2. Validate the suggested changes. The SQL script is generated. You can still view the SQL script to eliminate some SQL statements and ask for the regeneration of the script.
3. Proceed with the execution of the script.

Comparing Change Propagation and 11g SQL Scripts

| Category | Change Propagation | SQL Scripts |
|--|--------------------|-----------------------|
| Intelligent validation of changes | Yes | No |
| Preview and edit changes before apply | Yes | No |
| Automatically save and version changes | Yes | No |
| Display log of changes applied | Yes | Yes |
| Maintain history of change logs | Yes | No (else manually) |
| Centrally managed | Yes | No |
| Allows execution without revealing database passwords | Yes | No |
| Designed to handle database changes across multiple database environments | Yes | No |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

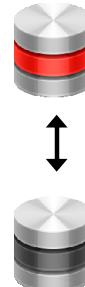
Change Propagation Benefits Using Change Management Pack Rather than SQL Scripts

- Change propagation with Change Management Pack provides an intelligent validation of changes. The method using 11g SQL scripts is time-consuming and potentially error prone.
- The comparison allows you to preview and edit the changes. The changes are automatically identified. It is easier to handle all the changes suggested after comparison. You just have to evict or accept them.
- You can keep track of the different versions of the changes through the GUI Enterprise Manager tool. This is more difficult to handle this with scripts: you have to manually create a directory to host the spool file and to maintain a history of change logs.
- The changes are centrally stored in Enterprise Manager repository.
- The passwords are not exposed as this is the case with SQL scripts. Enterprise Manager provides this capability to store credentials that are very safe regarding secure identification as the developers and DBAs don't have to remember passwords to create baselines in a development database or apply changes in a target database.
- It is designed to handle database changes application across multiple database environments such as development, test, reporting, production databases again and again. The changes are captured in one place and can then be applied in several targets databases.

Database Lifecycle Management Pack Schema Change Plans

Change plans allow users to specify, group, and package object metadata changes.

- Create change plans from
 - *Ad hoc* changes
 - Comparison-based differences
 - Developer tools
- Apply changes to multiple targets.



Role-based workflow

- **Developer:** Create and store change plan via SQL Developer.
- **DBA:** Review / apply change plan.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The application developers need an automated mechanism to capture the development changes to hand off to the DBA instead of sending scripts back and forth which is time-consuming and potentially error prone process. Schema Change Plan provides the automated way for developers and DBAs to work on application schema changes via a common and single object, the change plan.

This object is created by developers to collect and store application schema changes, which is reviewed and applied into any target system by DBAs. This is therefore a container object for change requests.

According to your role in the enterprise, you use different tools to manage the change plans.

1. The developer creates and stores a change plan with SQL Developer tool in Enterprise Manager repository. These changes can be either ad hoc changes, such as newly created tables and indexes, added or dropped columns in tables, modified procedures, added triggers, modified views. The DBA can also create change plans relying on comparison-based changes. For example, if you compare two databases that have a common table, but in one database, the table contains an additional column that the same table in the other database does not have, then this is a change that can be included in a change plan. These changes are traditionally manually collected in a script. Enterprise Manager 12c Database Lifecycle Management introduces this new object, the change plan in which developers or DBAs store all application changes. Change plans are not associated with a source database.

2. Using Enterprise Manager GUI tool, the DBA reviews the changes within the change plan, edit the changes and validate them or not, and finally requests the SQL script generation. When the script generation is completed, the DBA executes the SQL script to apply all the accepted changes to several databases.

If developers are not allowed to access the test or production environment, the key thing introduced is the ability to work within a role-based workflow. If you look at the process at the bottom of the slide, the developer and DBA have different roles. The developer uses SQL Developer to manage the application development providing some schema changes. He creates a change plan to store changes and stores it in Enterprise Manager repository.

Then, the DBA uses Enterprise Manager to review and edit the schema change plan. After he validated the changes, he requests the SQL script generation and executes it at a scheduled time to apply the changes to multiple targets.

Change Requests

- A change plan contains change requests for one or more metadata objects.
- A change request can be a request to:
 - Create an object
 - Drop an object
 - Modify one or more attributes of an object
- When a change plan is deployed to synchronize the target:
 - The change is analyzed in the context of the database being deployed
 - A relevant PL/SQL script is generated, based on the metadata at the target database



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A schema change plan contains change requests for one or more metadata objects. A change request can take three forms:

- Creates a database object: it contains the complete definition of the database object. For example, there is a new table that has been created in a development database and that needs to be added in the target database application.
- Drops a database object: it consists of the type, schema and name of the object, and an instruction to drop that object. For example, a table or an index does not need to exist anymore in the target database because the developer dropped it from the application.
- Modifies a database object: it contains one or more instructions pertaining to specific object attributes. These can take various forms:
 - Modify the value of an attribute: for example, change a table's tablespace to USERS or the data type of column C1 from VARCHAR2 (50) to VARCHAR2 (100)
 - Add or remove a sub-object, such as a table column or partition. For example, modify one or more attributes of an object if some columns need to be added or dropped in the table in the target database.
 - Change the value of an initialization parameter

An object can be monitored so that in case any change occurs on this object, the change plan will be informed of the type of change applied on this monitored object.

A schema change plan is not a SQL script. The change requests consist of abstract specifications of the changes to be made. Deploying a change plan to a database results in a script that is appropriate for that database.

When a schema change plan is validated, what should also be considered during the deployment phase, is that the changes are analyzed in the context of the target database to determine whether the changes can be carried out at all. For example, if a change suggests to add a column in the target database table in comparison to the same table from the test database, and if the column, meanwhile, had already been manually added to the target table, the analysis reports that it is useless to apply the change. Another example, if a change request seeks to modify a table that does not exist in the destination database, it cannot be done. Similarly, if a change request adds a foreign key constraint to a table and the constraint references a table that does not exist, the constraint cannot be added to the table.

It determines whether the change may have harmful consequences. For example, if a table column has the data type VARCHAR2 (200), a request to change the data type to VARCHAR2 (100) may result in script execution failure or truncation of existing data.

It creates the DDL statements needed to carry out the change request.

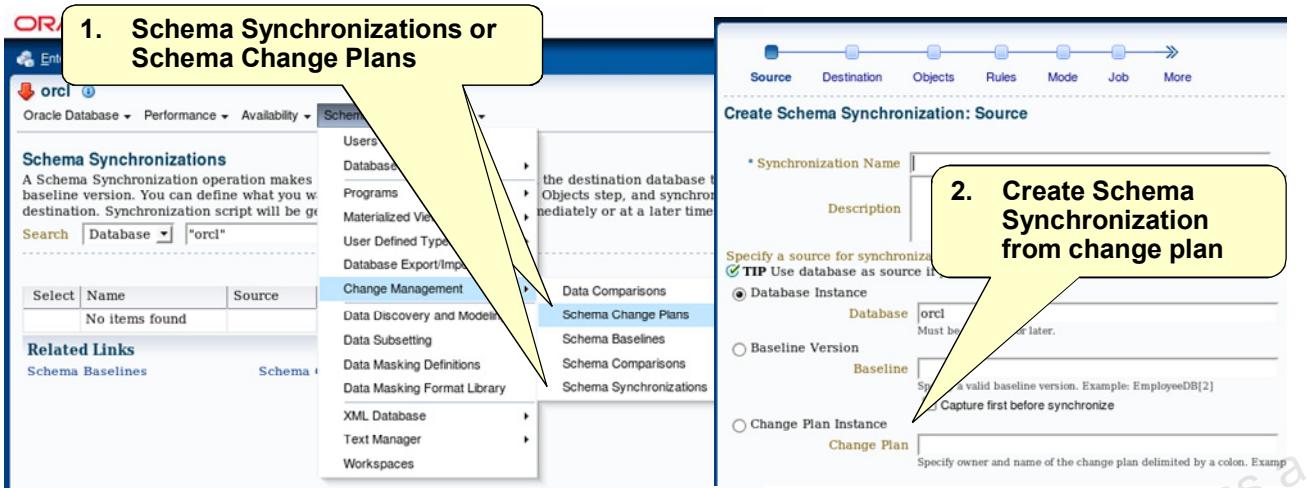
It determines the dependencies between objects affected by the schema change plan deployment and orders the DDL statements appropriately.

This is an intelligent analysis which validates the changes made in the test database against the actual definitions that are in the target database to check if the changes can be applied successfully.

It generates the Impact Report indicating when a change request cannot be carried out or may result in a problem. The DBA can examine the Impact Report, along with the generated script, before executing the script.

The PL/SQL script is generated for the specific target database, based on the metadata at the target database and this step still allows the DBA to exclude certain actions to be performed on the target database before execution takes place.

Schema Synchronization



- If Schema Synchronizations, then Create Schema Synchronization from
 - “Database” or “Baseline” or “Change plan”
- If Schema Change Plans, then “Create Synchronization from Change Plan” selected

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Schema change plans complement the existing Change Management Dictionary Synchronization component as it is a new source for synchronization.

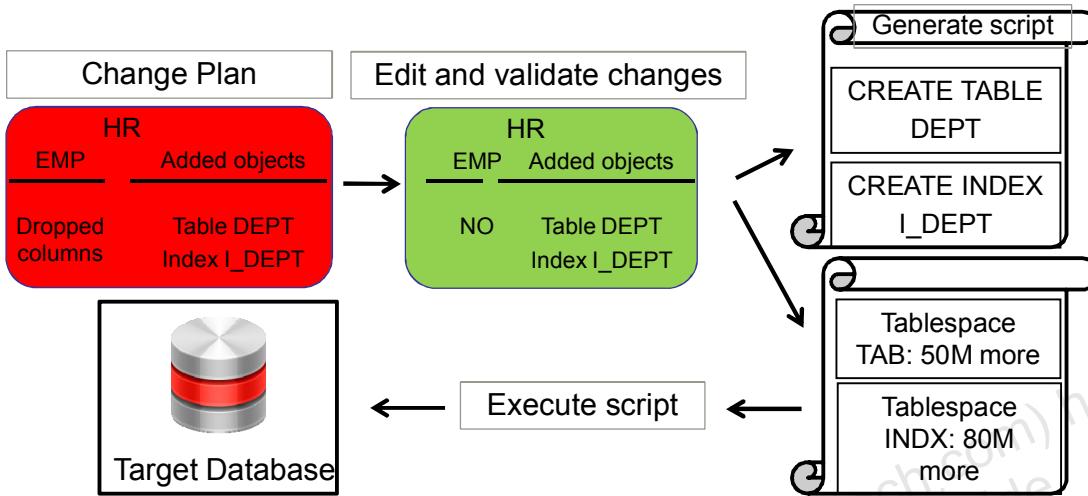
Schema Synchronization operation makes the definitions of set of objects in the destination database the same as those in the source and is still performed in three steps.

The step before synchronization is still the comparison of the source with the target database: but this source can be either a database or a baseline version baseline or a change plan. In previous Oracle Change Management pack version, the source for comparison was restricted to either a baseline or a source database. The DBA just needs to review all changes contained in the schema plan changes, validate them before requesting the script generation.

You can perform the first step of the synchronization in two ways:

- Click on “Change Management” option in the “Schema” menu, then “Schema Synchronizations” option and then “Create Schema Synchronization” where you will be prompted to choose the source of the synchronization. In Enterprise Manager Cloud Control, you now have the change plan as a possible source.
- Click on “Change Management” option in the “Schema” menu, then “Schema Change Plans” option, and select the change plan amongst the available ones to synchronize a target with. Then click the “Create Synchronization from a Change plan” button. The analysis is performed to display the inconsistencies in the change requests. The generation of the script still allows you to exclude certain actions to be performed on the target database before execution takes place.

Schema Synchronization



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The step before synchronization is still the comparison of the source with the target database where the source can be now a change plan. The DBA reviews all changes issued after the comparison, validates them before requesting the script generation.

After the “Create Synchronization from a Change Plan” analysis is performed, it displays the inconsistencies in the change requests. The generation of the script still allows you to exclude certain actions to be performed on the target database before execution takes place.

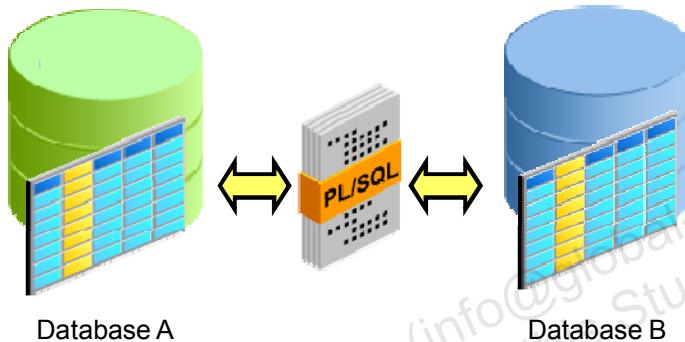
Database Lifecycle Management Pack

Data Comparisons

Schema Change
Plans
**Data
Comparisons**

Data Comparisons fills a critical gap to allow:

- Application vendors to compare seed data
- Application customers to compare configuration data between different sites
- DBAs to determine how seed data customizations will be affected by application upgrades



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Data Comparisons of Database Lifecycle Management pack fills a critical gap in the Change Management Pack lacking this capability of data comparison.

Application vendors need to compare seed data between application versions.

Customers of applications may want to compare their configuration data between different sites.

DBAs may want to know how their seed data customizations will be affected during the upgrade of the application database. This can be done by cloning the production database to a test database, upgrading the test database and then comparing the seed data between test and production.

There are a number of such applications of data comparisons.

Data Comparisons is a new feature added to the Database Lifecycle Management Pack. This is accessible by Enterprise Manager Cloud Control GUI pages that provide an interface to access the `DBMS_COMPARISON` package, available in the Oracle Database.

DBMS_COMPARISON

Data Comparisons provides a GUI interface to access the DBMS_COMPARISON package.

- Compares data between a referenced and a candidate database
 - Requires a database link between the two databases
 - Referenced and candidate can be the same database
- Can be used for different types of data:
 - Seed data – Provided with an application on installation
 - Configuration data – Parameters for the application that are set up by the user
 - Master data – Data families that are of interest to the business (for example, customers, suppliers, products, employees and so on)
 - Transaction data – Records the operation of business processes



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Data Comparisons is dependent on the DBMS_COMPARISON package of Oracle Database. DBMS_COMPARISON is a package appearing in Oracle Database 11g that enables to compare database objects in different databases and identify differences between them.

DBMS_COMPARISON compares data in database objects between a referenced database and a candidate database.

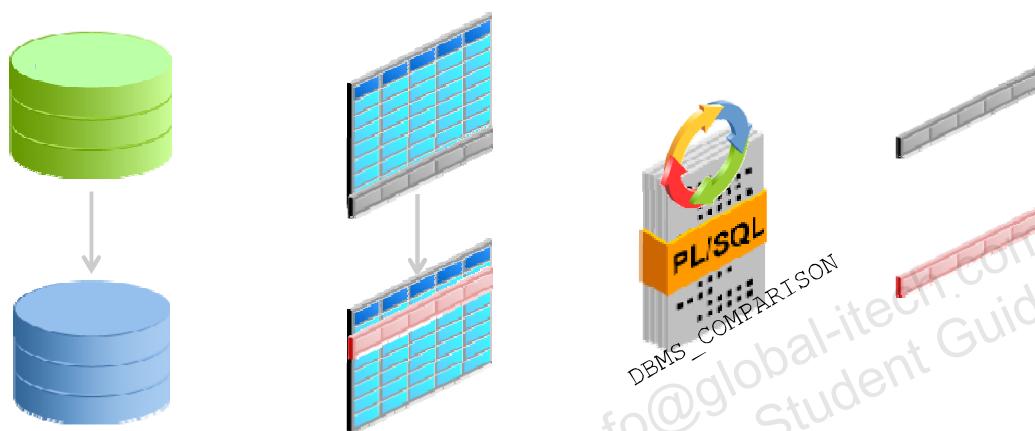
Referenced database is the one that runs the subprograms in the DBMS_COMPARISON package. Be advised that the referenced database carries an additional processing load and requires some space to store the row IDs of differing rows (not the entire rows themselves). If you compare data between a production system and a test system, it might be appropriate to process and store the results on the test system.

An object in the referenced database is compared with one in the candidate database.

The referenced and candidate databases can be one and the same. Otherwise, a database link created in the referenced database needs to be passed to DBMS_COMPARISON to allow it to access data in the candidate database.

- Seed data that is required by the application to function is data that comes with the application on installation. It can be provided with the system for training, testing, or as template for the data that user enters. For example, Siebel ships with seed data in a List of Values (LOV) definition, default mappings of views to responsibilities, and predefined queries.
- Configuration data consists of parameters for the application that are set up by the user. Depending on the kind of application, configuration data may include a chart of accounts, a supplier's payment terms, or a list of diagnostic codes.
- Master data refers to families of data that are of interest to the business (customers, suppliers, products, employees). These are typically the resources of the business and they fit into subject categories. All businesses, regardless of the type of business, store data about their people, products, financial, information, and physical resources.
- Transaction data records the operation of the business processes. Examples of transactions include orders, invoices, or payment records. This type of data is expected to grow quickly and form the bulk of data in transaction processing databases.

Flow



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The flow is simple.

1. You create a data comparison between two databases or within the same database.
2. You define the objects for which data comparison is needed.
3. You submit a job to perform the comparison
4. View the results to verify if there are any difference in terms of data between the objects compared.

Guidelines

- Referenced database must be of Oracle Database 11g Release 1 or later, candidate database must be Oracle Database 10g Release 1 or later.
- Database character sets must be the same.
- Data can be compared for tables, single-table views, materialized views and synonyms.
- An index must uniquely identify rows in both objects.
- Data cannot be compared for some data types (for example, LONG, LONG RAW, ROWID, CLOB, BLOB).
 - These columns can, however, be excluded from the comparison.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The referenced database always executes the comparison, therefore the Oracle Database version must be Oracle Database 11g or later. The candidate database for comparisons must be of Oracle Database 10g or later. It is either a remote database accessed by a database link or the same database to compare different schema objects.

The database character sets must be the same for the referenced and candidate databases.

Data Comparisons allows you to compare tables, single-table views, materialized views, and synonyms for tables. To compare rows from a referenced object with rows of a candidate object, the rows must be uniquely identified. A primary key constraint or a unique constraint on one or more non-NULL columns can satisfy this requirement.

So to compare rows from a referenced object with rows of a candidate object, the database objects must have one of the following types of indexes:

- A single-column index on a number, time stamp, interval, or DATE data type column
- A composite index that only includes number, time stamp, interval, or DATE data type columns. Each column in the composite index must either have a NOT NULL constraint or be part of the primary key.

In case these constraints are not present on a table, then user needs to create an index whose columns satisfy this requirement. In case this requirement is not satisfied, you get an error message while viewing the results such as:

```
ORA-23626: No eligible index on table SCOTT.BONUS ORA-06512: at  
"SYS.DBMS_COMPARISON", line 5002 ORA-06512: at  
"SYS.DBMS_COMPARISON", line 448 ORA-06512: at line 2
```

Most data types are considered during comparisons between two objects. But certain data types are not supported during comparison. Examples LONG, LONG RAW, ROWID, CLOB, BLOB. These columns can, however, be excluded from the comparisons.

Creating a Data Comparison

1. Create a data comparison.
 - Reference database
 - Candidate database
2. Select object pairs or add multiple objects to the list.

| Reference Schema | Reference Object | Candidate Schema | Candidate Object | Included Columns |
|------------------|------------------|------------------|------------------|------------------|
| HR | DEPARTMENTS | HR_BIS | DEPARTMENTS | |

3. Submit comparison job.
4. View comparison results.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

From the Enterprise menu, choose a target database and then in a database context, from the Schema menu, click Change Management and then Data Comparisons.

1. Create the data comparison: Provide the names of the referenced and candidate databases that can be either different databases or the same database. The referenced database is the one that contains the objects of reference.
2. Through the “Actions” menu, define object pairs by specifying the referenced and candidate objects. If the referenced database is the same as the candidate database, the objects are from the same database. In the example on the slide, the comparison is defined between the HR . DEPARTMENTS table from HR database with the HR_BIS . DEPARTMENTS table from SALES database. Select one or more columns of the referenced or candidate objects for comparison being common to both objects. Optionally select an index used for comparison if several exist, specify a where clause per pair of objects being compared, and configure the comparison to a point in time. You declare object pairs for comparison or multiple objects. Adding multiple objects enables a bulk inclusion of multiple objects from the referenced database into the specification. You can search and select multiple objects, such as tables and views from the referenced database, and then edit each item as needed.
3. Then submit the comparison job.
4. When the comparison job has completed, view the results of the comparison.

Comparison Job and Results

Data Comparisons

A data comparison operation compares objects in a candidate database with objects in a reference database. To compare objects in a candidate database with objects in a reference database, you must first create a comparison job. A comparison job defines the objects to be compared and specifies the comparison criteria. When you run a comparison job, Data Comparisons compares the objects in the candidate database with the objects in the reference database. You can run a comparison job immediately or at a later time. You can also cancel a comparison job. Comparison results are generated when you run a comparison job. The results will be deleted when you delete the comparison job.

Comparison jobs

Submit Comparison job.

| Name | Reference Database | Candidate Database | Comparison Start Date | Job Status | Owner | Description |
|--------------|--------------------|--------------------|-----------------------|--------------------|----------|-------------|
| My_compare | hr.example.com | sales.example.com | 2011-06-23 at 09:32 | Completed with Err | DB_ADMIN | |
| My_compare_3 | hr.example.com | sales.example.com | 2011-06-23 at 09:43 | Succeeded | SYSMAN | |
| My_compare_2 | hr.example.com | sales.example.com | 2011-06-23 at 09:34 | Succeeded | SYSMAN | |

Data Comparison Results: My_compare_3

View row differences between

There are differences.

| View | View Row Differences | Detach | | | | | |
|------------------|----------------------|------------------|------------------|--------|---------------------|---------------------|--------------------|
| Reference Schema | Reference Object | Candidate Schema | Candidate Object | Result | Reference Only Rows | Candidate Only Rows | Non-identical Rows |
| HR | DEPARTMENTS | HR_BIS | DEPARTMENTS | ≠ | 14 | 1 | 1 |

Reference-only rows
Candidate-only rows
Non-identical rows

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The first screenshot displays the comparison jobs that were submitted. Three comparisons were performed with different configurations.

The second screenshot shows the result of one of the completed comparisons My_Compare_3. Data Comparisons attempts to compare all tables. Look for rows in the “Result” column with the ≠ symbol, indicating that there are differences between referenced row and candidate row data. If the symbol is = then all rows are identical according to the selected columns for Comparisons. If for any reason (for example a missing unique index on any of the involved columns), the comparison failed, you can view the error messages by selecting the Messages tab that is not displayed on this window but would be at the bottom of the window. An error message is indicated with an red X instead of the ≠ symbol.

You can also view the SQL statements executed to perform the comparisons by clicking the Executed Statements tab. You would see the following detailed statements such as:

- create database link that creates the link between the referenced database and the candidate database
- dbms_comparison.create_comparison that creates a named comparison with the list of objects to be compared
- dbms_comparison.compare that executes the comparison

If you want to view the details of the differences between two objects, click View Row Differences.

Results: Reference Only Rows

- Reference-only rows
- Candidate-only rows
- Non-identical rows

The screenshot shows the 'Row Data Differences' interface for comparing two databases: 'hr.example.com' (Reference Database) and 'sales.example.com' (Candidate Database). Both are logged in as 'system'. The comparison is based on the 'HR.DEPARTMENTS' object, with the index column being 'DEPARTMENT_ID'. A green box highlights the 'Reference-only rows' section, which includes a checked checkbox for 'Reference Only Rows'. Below this, a note states: 'Rows are categorized based on their index column values. Reference only and candidate only rows are those with index column values present on one side respectively. Non-identical rows are those with index column values present on both sides, but with one or more different values.' A yellow box labeled 'Reference object' points to the first row of the data grid, which is colored blue. A green box on the right contains the text: 'All rows existing in referenced object but not in candidate object'. The data grid shows six rows, all of which are blue ('Reference') and have 'DEPARTMENT_ID' values 10, 140, 150, 160, 170, and 180.

| Row Source | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|------------|---------------|----------------------|------------|-------------|
| Reference | 10 | Administration | 200 | 1700 |
| Reference | 140 | Control And Credit | | 1700 |
| Reference | 150 | Shareholder Services | | 1700 |
| Reference | 160 | Benefits | | 1700 |
| Reference | 170 | Manufacturing | | 1700 |
| Reference | 180 | Construction | | 1700 |

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Row Data Differences page allows you to view:

- Reference-only rows
- Candidate-only rows
- Non-identical changed rows
- All of them

The Row Source column indicates the origin of each row of data as a whole. Furthermore, data in a row differing between referenced and candidate are displayed in contrasting colors, indicating whether the source of the data comes from the referenced object (in blue) or the candidate object (in red).

The comparison is shown based on a key column (depending on a chosen unique index). If the key column value is different, the row appears as a candidate-only or reference-only row. If other columns are different, the row appears as a non-identical row.

In the example in the slide, you chose to view the rows that exist with index column values in the referenced object but not in the candidate object. For example, the rows with DEPARTMENT_ID 10, 140 do not exist in the HR_BIS.DEPARTMENTS table.

Results: Candidate-Only Rows

- Reference-only rows
- Candidate-only rows
- Non-identical rows

Row Data Differences: My_compare_3

Reference Database: hr.example.com Candidate Database: sales.example.com
 Logged in As system Logged in As system
 Reference Object: HR.DEPARTMENTS Candidate Object: HR_BIS.DEPARTMENTS
 Index Columns: DEPARTMENT_ID

Show: Reference Only Rows Candidate Only Rows Non-identical Rows

Rows are categorized based on their index column values. Reference only and candidate only rows are those respectively. Non-identical rows are those with index column values present on both sides, but with one or more differences.

| View | Detach | | | |
|------------|---------------|-----------------|------------|-------------|
| Row Source | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
| Candidate | 280 | NEW_Payroll | | 1700 |

Candidate - only rows

All rows existing in candidate object but not in referenced object



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, you chose to view the candidate-only rows, those with index column values that exist only in the candidate object but not in the referenced object. For example, the rows with DEPARTMENT_ID 280 does not exist in the referenced object HR.DEPARTMENTS whereas it does in the candidate object HR_BIS.DEPARTMENTS.

Results: Non-Identical Rows

- Reference-only rows
- Candidate-only rows
- Non-identical rows between **referenced object** and **candidate object**

Row Data Differences: My_compare_3

Reference Database: hr.example.com Candidate Database: sales.example.com
 Logged in As: system Logged in As: system
 Reference Object: HR.DEPARTMENTS Candidate Object: HR_BIS.DEPARTMENTS
 Index Columns: DEPARTMENT_ID

Show: Reference Only Rows Candidate Only Rows Non-identical Rows

Rows are categorized based on their index column values. Reference only and candidate only rows are respectively those index column values present on one side, but not the other.

Non-identical rows

| Row Source | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|------------|---------------|------------------|------------|-------------|
| Reference | 240 | Government Sales | 1700 | |
| Candidate | 240 | Government Sales | 1800 | |

All rows existing in both referenced and candidate objects but with different values

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the example on the slide, you chose to view the non-identical rows. This means those with index column values present on both sides, but with one or more differences in the other (non-index) column values. In this example, the row with DEPARTMENT_ID 240 exists in both objects HR.DEPARTMENTS and HR_BIS.DEPARTMENTS but with LOCATION_ID 1700 in the reference object and 1800 in the candidate object.

Quiz

The schema change plan in the target database is used to:

- a. Modify one or more attributes of an object
- b. Create a missing object
- c. Drop an object
- d. All of the above



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: d

Quiz

The Data Comparisons job shows the following differences between two compared objects:

- a. Reference-only rows
- b. Candidate-only rows
- c. Non-identical rows between referenced object and candidate object
- d. All of the above



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: d

Summary

In this lesson, you should have learned how to explain:

- Why and when to use the new Schema Change Plans feature
- How to use schema change plans
- Why and when to use the new Data Comparisons feature
- How to use data comparisons

Practice 14 Overview: Schema Change Plans and Data Comparisons

This practice covers the topic of Schema Change Plans using a demonstration.



Module

Performance



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

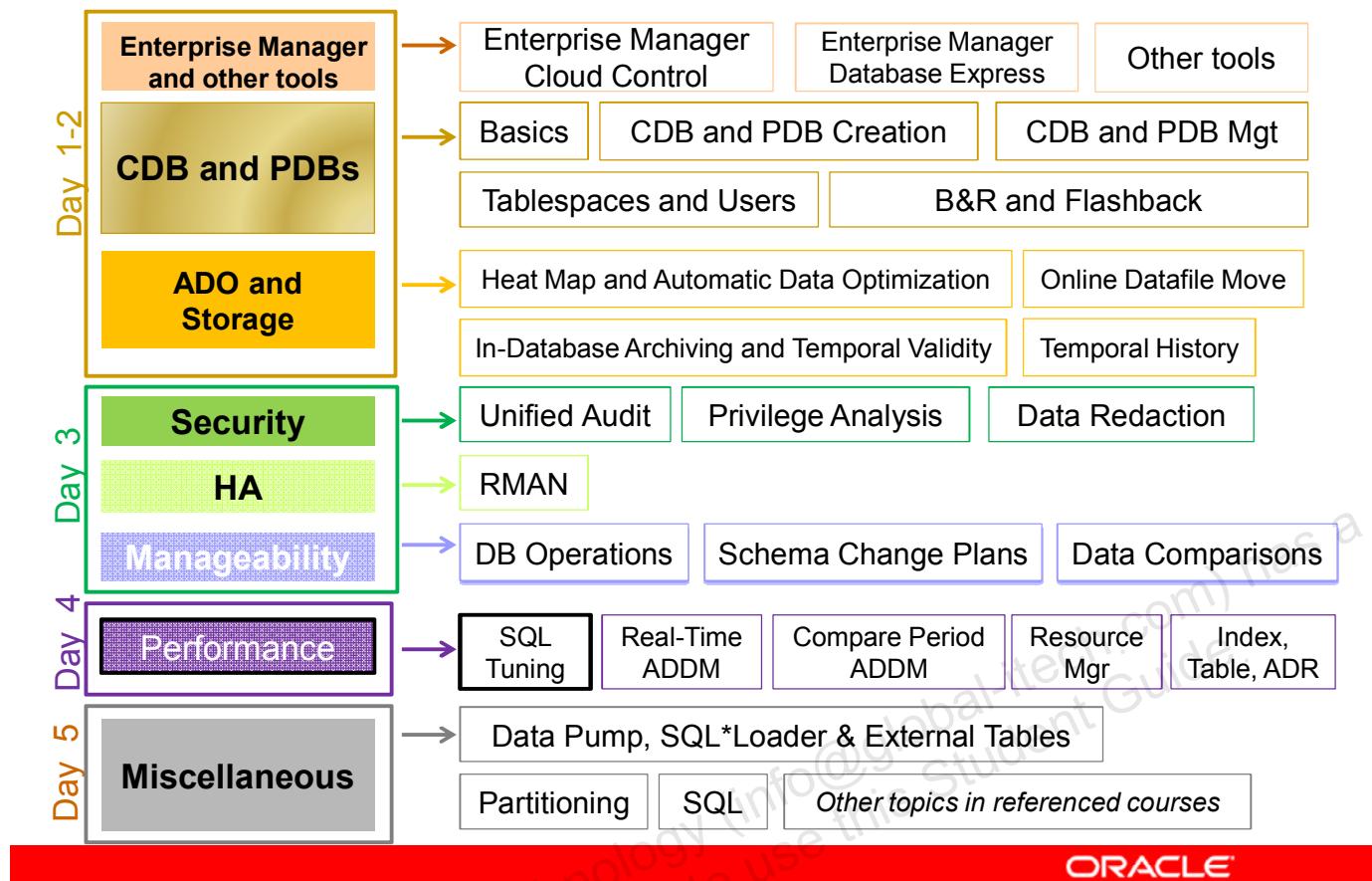
15

SQL Tuning Enhancements

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several optimizer and statistics enhancements, and a new SQL plan directives feature.

Objectives

After completing this lesson, you should be able to:

- Describe Adaptive SQL Plan Management
- Describe enhancements to the SQL management base
- Use a dynamic plan to improve query performance
- Describe re-optimization for adaptive execution plans
- Use SQL plan directives to generate a better plan
- Describe statistics gathering performance improvements
- Use new histograms
- Describe enhancements to extended statistics and detect useful column groups for a specific workload
- Describe enhancements to dynamic sampling

A solid red horizontal bar at the bottom of the slide.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Refer to other sources of information:

- *Oracle Database 12c New Features Demo Series* demonstrations under Oracle Learning Library.

Road Map

- Adaptive SQL Plan Management
- Adaptive Execution Plans
- Optimizer Statistics Management

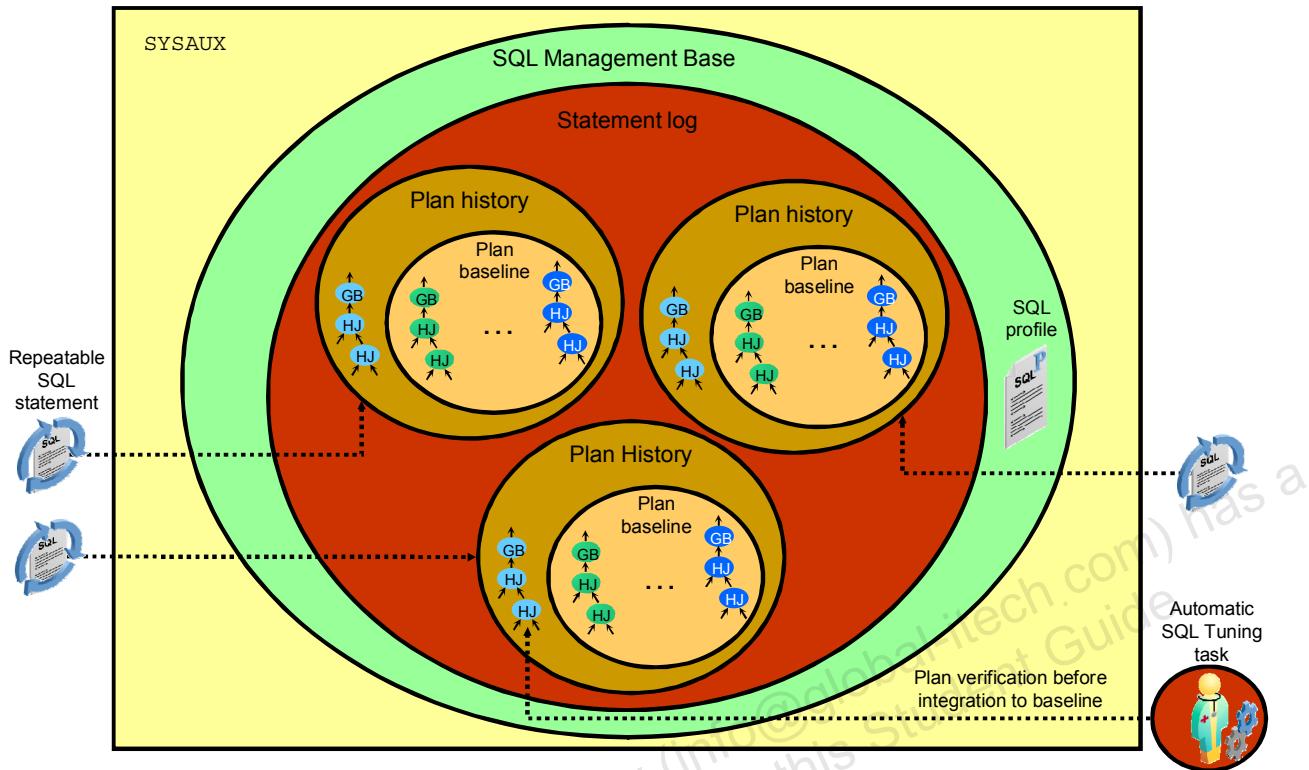


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This lesson consists of three major parts:

- Adaptive SQL Plan Management
- Adaptive Execution Plans
- Optimizer Statistics Management

SQL Plan Baseline: Architecture



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SQL Plan Management (SPM) feature introduces necessary infrastructure and services in support of plan maintenance and performance verification of new plans.

For SQL statements that are executed more than once, the optimizer maintains a history of plans for individual SQL statements. The optimizer recognizes a repeatable SQL statement by maintaining a statement log. A SQL statement is recognized as repeatable when it is parsed or executed again after it has been logged. After a SQL statement is recognized as repeatable, various plans generated by the optimizer are maintained as a plan history containing relevant information (such as SQL text, outline, bind variables, and compilation environment) that is used by the optimizer to reproduce an execution plan.

As an alternative or complement to the automatic recognition of repeatable SQL statements by setting the parameter `optimizer_capture_sql_plan_baselines` to TRUE, manual seeding of plans for a set of SQL statements is also supported.

A plan history contains different plans generated by the optimizer for a SQL statement over time. However, only some of the plans in the plan history may be accepted for use. For example, a new plan generated by the optimizer is not normally used until it has been verified not to cause a performance regression. Plan verification is done “out of the box” as part of Automatic SQL Tuning running as an automated task in a maintenance window.

An Automatic SQL Tuning task targets only high-load SQL statements. For them, it automatically implements actions such as making a successfully verified plan an accepted plan. A set of acceptable plans constitutes a SQL plan baseline. The very first plan generated for a SQL statement is obviously acceptable for use; therefore, it forms the original plan baseline. Any new plans subsequently found by the optimizer are part of the plan history but not part of the plan baseline initially.

The statement log, plan history, and plan baselines are stored in the SQL Management Base (SMB), which also contains SQL profiles. The SMB is part of the database dictionary and is stored in the SYSAUX tablespace. The SMB has automatic space management (for example, periodic purging of unused plans). You can configure the SMB to change the plan retention policy and set space size limits.

Note: If the database instance is up but the SYSAUX tablespace is OFFLINE, the optimizer is unable to access SQL management objects. This can affect performance on some of the SQL workload.

SQL Plan Management: Overview

SQL Plan Management ensures that runtime performance never degrades due to the change of an execution plan.

Optimizer automatically manages “execution plans.”

- Only known plans are used.
- Plans can be manually or automatically captured.

Plan changes are verified.

- Only better-performing plans are used.
- Manual verification is performed either through EM or by running DBMS_SPM.EVOLVE_SQL_PLAN_BASELINES in Oracle Database 11g.
- Automatic verification is performed by nightly Automatic SQL Tuning job running DBMS_SPM.EVOLVE_SQL_PLAN_BASELINES in Oracle Database 11g.

SPM consists of the following components:

- SQL plan baseline capture
- SQL plan baseline selection
- SQL plan baseline evolution



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

SQL Plan Management (SPM) ensures that runtime performance never degrades due to the change of an execution plan. It has two main objectives:

- It prevents performance regressions resulting from sudden changes to the execution plan of a SQL statement by providing components for capturing, selecting, and evolving SQL plan information.
- It offers performance improvements by adapting to database system changes.

With SPM, the optimizer automatically manages execution plans and ensures that only known or verified plans are used. When a new plan is found for a SQL statement, the plan is not used until the database verifies that its performance is comparable to or better than the current plans. Verification is a manual process performed either through Oracle Enterprise Manager (EM) or by running DBMS_SPM.EVOLVE_SQL_PLAN_BASELINES in Oracle Database 11g.

Verification is an automatic process performed by the Automatic SQL Tuning (AST) job running DBMS_SPM.EVOLVE_SQL_PLAN_BASELINES in Oracle Database 11g.

The components of SPM are SQL plan baseline capture, SQL plan baseline selection and SQL plan baseline evolution. There are two ways to seed or populate a SQL management base (SMB) with execution plans:

- Automatic capture of execution plans (available starting with Oracle Database 11g)
- Bulk load execution plans or pre-existing SQL plan baselines

Adaptive SQL Plan Management

- The new evolve auto task runs in the nightly maintenance window.
- It ranks all non-accepted plans and runs the evolve process for them. Newly found plans are ranked the highest.
- Poor performing plans are not retried for 30 days and then only if the statement is active.
- The new task is `SYS_AUTO_SPM_EVOLVE_TASK`.
- Information about the task is in `DBA_ADVISOR_TASKS`.
- Use `DBMS_SPM.REPORT_AUTO_EVOLVE_TASK` to view the results of the auto job.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Starting in Oracle Database 12c, the database can use adaptive SPM. With adaptive SQL plan management, DBAs no longer have to manually run the verification or evolve process for non-accepted plans. They can go back days or weeks later and review what plans were evolved during each of the nightly maintenance window. When automatic SQL tuning is in `COMPREHENSIVE` mode, it runs a verification or evolve process for all SQL statements that have non-accepted plans during the nightly maintenance window. All the non-accepted plans are ranked and the evolve process is run for them. The newly found plans are ranked the highest.

If the non-accepted plan performs better than the existing accepted plan (or plans) in the SQL plan baseline, then the plan is automatically accepted and becomes usable by the optimizer.

After the verification is complete, a persistent report is generated detailing how the non-accepted plan performs compared to the accepted plan performance.

As the evolve process is now an `AUTOTASK` DBAs can also schedule their own evolve job at end time and use `DBMS_SPM.REPORT_AUTO_EVOLVE_TASK` to view the results of auto job.

The Automatic Plan Evolution of SQL Plan Management is a feature of the Tuning Pack.

Automatically Evolving SQL Plan Baseline

SPM Evolve Advisor

- The evolve function is now an advisory task.
- The new function allows DBAs to create a task, evolve plans in the task, and so on.

DBMS_SPM task-based functions

- DBMS_SPM.CREATE_EVOLVE_TASK
- DBMS_SPM.EXECUTE_EVOLVE_TASK
- DBMS_SPM.REPORT_EVOLVE_TASK
- DBMS_SPM.IMPLEMENT_EVOLVE_TASK

Persistent plan evolution report

- Allows the advisory task infrastructure to be implemented any time.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

SPM Evolve Advisor is a SQL advisor that evolves plans that have recently been added to the SQL plan baseline. The advisor simplifies plan evolution by eliminating the manual chore of evolution.

New functions introduced in the DBMS_SPM package work on the advisory task infrastructure. The new functions such as DBMS_SPM.CREATE_EVOLVE_TASK, DBMS_SPM.EXECUTE_EVOLVE_TASK, and so on allow database administrators (DBAs) to create a task, evolve one or more plans in the task, fetch the report of the task, and implement the results of a task. The task can be executed multiple times, and the report of a given task can be fetched multiple times. SPM can schedule an evolve task as well as rerun an evolve task. There is also a function to implement the results of a task, which in this case would be to accept the currently non-accepted plans.

Using the task infrastructure allows the report of the evolution to be persistently stored in the advisory framework repository. It also allows Enterprise Manager to schedule SQL plan baselines evolution and fetch reports on demand. The new function supports HTML and XML reports in addition to TEXT. In Oracle Database 12c, the new task-based functions in DBMS_SPM retain the time_limit specification as a task parameter. It has the same default values as Oracle Database 11g. As for commit, the new evolve task does not accept any plans by default. Users can view the report by using the appropriate function, and then execute a new implement function to accept the successful plans.

SQL Management Base Enhancements

- Prior to Oracle Database 12c,
DBMS_XPLAN.DISPLAY_PLAN_BASELINE displayed the execution plan by compiling the statement with the baseline. Pre-12c
- In Oracle Database 12c, when a new plan is added to the plan history of a SQL statement, plan rows are also stored in the SQL management base (SMB). When DBMS_XPLAN.DISPLAY_PLAN_BASELINE is executed, you fetch the plan data from the SMB. 12c
- Easier diagnosability when plan cannot be reproduced



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SQL management base resides in the SYSAUX tablespace and store SQL plan baselines, statement logs, plan histories, and SQL profiles.

In Oracle Database 11g, DBMS_XPLAN.DISPLAY_PLAN_BASELINE displayed the execution plan by compiling the statement with the baseline, mainly using outlines.

In Oracle Database 12c, when a new plan is added to the plan history of a SQL statement, plan rows are also stored in the SQL management base. When DBMS_XPLAN.DISPLAY_PLAN_BASELINE is executed, you fetch the plan data from the SQL management base.

In Oracle Database 11g, the SQL Plan Management (SPM) plan rows are not stored in the SMB. When customers upgrade to Oracle Database 12c, the plan rows are populated during execution of the plans.

Quiz

The `report_evolve_task` function is called to display the results of an evolve task.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Lesson Road Map

- Adaptive SQL Plan Management
- Adaptive Execution Plans
- Optimizer Statistics Management



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You now move on to adaptive execution plans.

Adaptive Execution Plans

- A query plan changes during execution because runtime conditions indicate that optimizer estimates are inaccurate.
- All adaptive execution plans rely on statistics that are collected during query execution.
- The two adaptive plan techniques are:
 - Dynamic plans
 - Re-optimization
- The database uses adaptive execution plans when `OPTIMIZER_FEATURES_ENABLE` is set to 12.1.0.1 or later, and `OPTIMIZER_ADAPTIVE_REPORTING_ONLY` initialization parameter is set to the default of FALSE.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Adaptive Execution Plans feature enable 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 which 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 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.

Dynamic Plans

- The final decision is based on statistics that are collected during execution.
- Alternate subplans are precomputed and stored in the cursor.
- Statistic collectors are inserted at key points in the plan.
- If statistics prove to be out of range, subplans can be swapped.
- It requires buffering near the swap point to avoid returning rows to users.
- Only join methods and the distribution method can change.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A dynamic plan is an execution plan that has different built-in plan options. During the first execution, before a specific subplan becomes active, the optimizer makes a final decision about which option to use. The optimizer bases its choice on observations made during the execution up to this point. A dynamic plan enables the final plan for a statement to differ from the default plan, thereby potentially improving query performance.

A subplan is a portion of a plan that the optimizer can switch to as an alternative at run time. During statement execution, the statistics collector buffers a portion of rows. Portions of the plan preceding the statistics collector can have alternate subplans, each of which is valid for a subset of possible values returned by the collector.

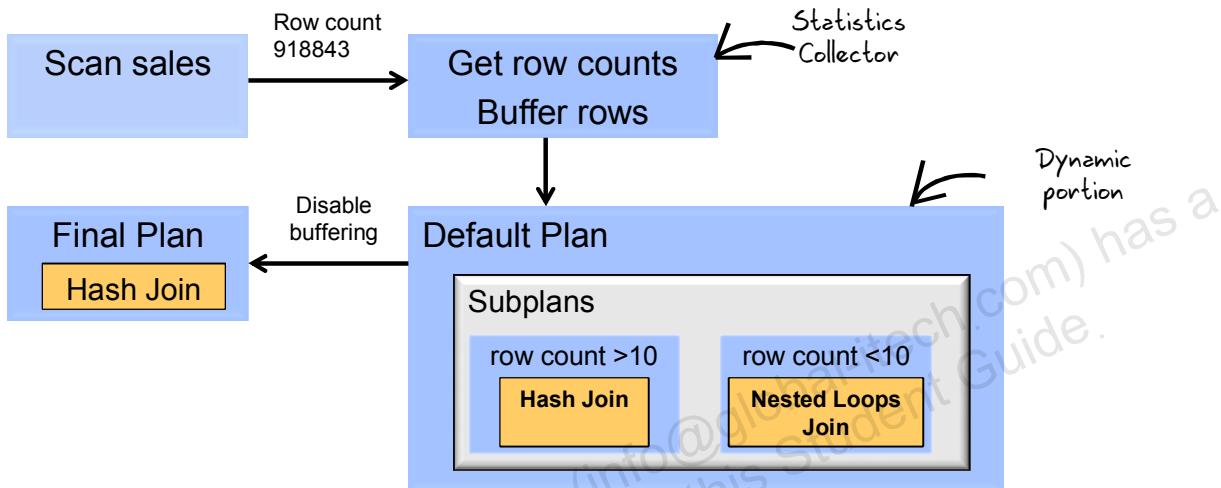
Often the set of values for a subplan is a range. If a statistic falls into the valid values of a subplan that was not the default plan, then the optimizer chooses the alternative subplan. After the optimizer chooses a subplan, buffering is disabled. The statistics collector stops collecting rows, and passes them through instead. On subsequent executions of the child cursor, the optimizer disables buffering, and chooses the same final plan.

With dynamic plans, the execution plan adapts to the optimizer's poor plan choices, and correct decisions can be made during the first execution.

Note: The new column in `V$SQL IS_RESOLVED_DYNAMIC_PLAN` indicates if the final plan was not the default plan. Information found via dynamic plans is persisted as SQL plan directives.

Dynamic Plan: Adaptive Process

```
SQL> SELECT s.cust_id, c.cust_last_name
  2  FROM sh.sales s, sh.customers c
  3 WHERE s.cust_id = c.cust_id;
```



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The code example shows a join of the sales and customers tables. A dynamic plan for this statement could show two possible plans, one with a nested loop and the other with a hash join.

The following graphic shows the adaptive process. For the query given in the slide, the dynamic portion of the default plan contains two subplans, each of which uses a different join method. The collector retrieves row count statistics, and also buffers up to 10 rows.

If the row count is below 10, then the optimizer chooses the nested loop; otherwise, the optimizer chooses the hash join. Because the row count in sales table is over 10, the optimizer chooses a hash join, stops buffering, and makes the final plan.

Dynamic Plans: Example

| A.1. The default plan show NLJ | | A.2. The final plan shows HJ | |
|---|--|--|--|
| <pre>SQL> explain plan for select /*Q10*/ product_name from order_items o, product_information p where o.unit_price = 15 and quantity > 1 and p.product_id = o.product_id; Explained. SQL> select * from table(dbms_xplan.display(format=>'basic +note')); </pre> | | <pre>SQL> select /*Q10*/ product_name from order_items o, product_information p where o.unit_price = 15 and quantity > 1 and p.product_id = o.product_id; PRODUCT_NAME ----- Screws <B.26.S> 13 rows selected. SQL> select * from table(dbms_xplan.display_cursor(format=>'basic +note')); PLAN_TABLE_OUTPUT -----</pre> | |
| <pre>PLAN_TABLE_OUTPUT -----</pre> | | <pre>EXPLAINED SQL STATEMENT: -----</pre> | |
| <pre>select /*Q10*/ product_name from order_items o, product_information p where o.unit_price = 15 and quantity > 1 and p.product_id = o.product_id</pre> | | <pre>Plan hash value: 2615131494</pre> | |
| <pre>-----</pre> | | <pre> Id Operation Name -----</pre> | |
| <pre> 0 SELECT STATEMENT 1 NESTED LOOPS 2 NESTED LOOPS 3 TABLE ACCESS FULL ORDER_ITEMS 4 INDEX UNIQUE SCAN PRODUCT_INFORMATION_PK 5 TABLE ACCESS BY INDEX ROWID PRODUCT_INFORMATION </pre> | | <pre> 0 SELECT STATEMENT 1 HASH JOIN 2 TABLE ACCESS FULL ORDER_ITEMS 3 TABLE ACCESS FULL PRODUCT_INFORMATION </pre> | |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A statement that has a dynamic plan could show two possible plans: one plan with a nested loop before execution and another with a hash join after execution. This information is displayed in the plan output table.

The Note section of the execution plan indicates whether the plan is dynamic: this is a dynamic plan.

Reoptimization: Cardinality Feedback

- Cardinality feedback was introduced in Oracle Database 11g, Release 2.
- Cardinality feedback is useful for queries where the data volume being processed is stable over time.
- During query execution optimizer estimates are compared to execution statistic: if they vary significantly then a new plan will be chosen for subsequent executions

Note: CURSOR_SHARING parameter is either EXACT or FORCE.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Cardinality feedback was introduced in Oracle Database 11g, Release 2. This feature does a very limited form of re-optimization, after the first execution is completed. It automatically improves plans for queries for which the optimizer estimate improper cardinalities in the plan. The optimizer misestimates cardinalities for a variety of reasons, such as missing or inaccurate statistics, or complex predicates. Cardinality feedback is not used for volatile tables. It is useful for queries where the data volume being processed is stable over time.

The optimizer may enable cardinality monitoring feedback for the cursor in the following cases: tables with no statistics, multiple conjunctive or disjunctive filter predicates on a table, and predicates containing complex operators for which the optimizer cannot accurately compute selectivity estimates.

The feature works as follows: During query execution, optimizer estimates are compared to execution statistics. If they vary significantly, a new plan is chosen for subsequent executions. Statistics are gathered for the actual data volume and data type seen during execution. If the original optimizer estimates vary significantly, the statement is hard parsed during the next execution by using the execution statistics. Statements are monitored only once. If they do not show significant differences initially, they do not change in the future. Only individual table cardinalities are examined. Information is stored only in the cursor and is lost if the cursor ages out.

Cardinality Feedback: Monitoring Query Executions

```
SQL> SELECT /*+ gather_plan_statistics */ product_name
  2  FROM    order_items o, product_information p
  3  WHERE   o.unit_price = 15
  4  AND     quantity > 1
  5  AND     p.product_id = o.product_id;
```

| Id | Operation | Name | Starts | E-Rows | A-Rows |
|----|-----------------------------|------------------------|--------|--------|--------|
| 0 | SELECT STATEMENT | | 1 | 13 | 13 |
| 1 | NESTED LOOPS | | 1 | 13 | 13 |
| 2 | NESTED LOOPS | | 1 | 4 | 13 |
| 3 | TABLE ACCESS FULL | ORDER_ITEMS | 1 | 4 | 13 |
| 4 | INDEX UNIQUE SCAN | PRODUCT_INFORMATION_PK | 13 | 1 | 13 |
| 5 | TABLE ACCESS BY INDEX ROWID | PRODUCT_INFORMATION | 13 | 1 | 13 |

Predicate Information (identified by operation id):

| |
|---|
| 3 - filter(("O"."UNIT_PRICE"=15 AND "QUANTITY">>1)) |
| 4 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID") |

Initial cardinality estimate is more than 10X off.

Predicate Information

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Execution of the query given in the code box is monitored. The query has a combination of an equality filter predicate and an inequality filter predicate which causes the misestimation in the cardinality of the `order_items` table after these filters are applied. The optimizer chooses the given plan where you can see from the execution statistics (estimation: E-Rows and actual: A-Rows columns in the plan) that the cardinality of `order_items` is underestimated.

Cardinality Feedback: Reparsing Statements

```
SQL> SELECT /*+ gather_plan_statistics */ product_name
  2   FROM order_items o, product_information p
  3 WHERE o.unit_price = 15
  4 AND quantity > 1
  5 AND p.product_id = o.product_id;
```

The screenshot shows the Oracle SQL Plan Monitor interface. It displays a query execution plan with the following details:

| Id | Operation | Name | Starts | E-Rows | A-Rows |
|----|-------------------|---------------------|--------|--------|--------|
| 0 | SELECT STATEMENT | | 1 | 1 | 13 |
| 1* | HASH JOIN | | 1 | 13 | 13 |
| 2 | TABLE ACCESS FULL | ORDER_ITEMS | 1 | 13 | 13 |
| 3 | TABLE ACCESS FULL | PRODUCT_INFORMATION | 1 | 288 | 288 |

Predicate Information (identified by operation id):

- 1 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")
- 2 - filter(("O"."UNIT_PRICE"=15 AND "QUANTITY">>1))

Note

- cardinality feedback used for this statement

Annotations with red boxes highlight the 'HASH JOIN' operation and the note section. A callout points from the 'Note' annotation to the text: 'Cardinality feedback appears in the notes section of the plan.' Another callout points from the 'A-Rows' column of the plan table to the text: 'Cardinality estimates are now correct and trigger the join method to change.'

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Execution statistics are used to reparse the statement during the second execution of the query in the code box. During the second execution, the actual cardinality is fed back to the optimizer. And this time the optimizer gets the cardinality estimate right, and the optimizer displays a different plan.

Automatic Re-optimization

- The optimizer automatically changes a plan during subsequent executions of a SQL statement.
- Join statistics are also included.
- Statements are continually monitored to see if statistics fluctuate over different executions.
- It works with adaptive cursor sharing for statements with bind variables.
- A new column is added in `v$SQL_IS_REOPTIMIZABLE`.
- Information found at execution time is persisted as SQL plan directives.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In re-optimization, the optimizer adapts a plan only on subsequent statement executions. Deferred optimization can fix any suboptimal plan chosen because of incorrect optimizer estimates, such as a suboptimal distribution method or an incorrect choice of degree of parallelism. At the end of the first execution, the optimizer has a more complete set of statistics, and this information can improve plan selection in the future. A query may be re-optimized several times, each time creating a larger and more precise set of optimizer adjustments.

Re-optimization fixes plan problems that dynamic plans cannot solve. Whereas dynamic plans help change relatively local parts of a plan, dynamic plans are not feasible for some global plan changes. For example, a query with an inefficient join order might cause suboptimal performance, but it is not feasible to adapt the join order during execution. In these cases, the optimizer automatically considers re-optimization. For re-optimization, the optimizer must decide which statistics to collect and when.

As of Oracle Database 12c, join statistics are also included. Statements are continually monitored to verify if statistics fluctuate over different executions. It works with adaptive cursor sharing for statements with bind variables. This feature improves the ability of the query processing engine (compilation and execution) to generate better execution plans.

The optimizer uses the statistics in the following ways:

- The statistics collectors and their associated optimizer estimates determine if automatic re-optimization is an option. If statistics differ from estimates by more than a threshold, the optimizer looks for a replacement plan.

- After the optimizer identifies a query as a re-optimization candidate, the database submits all collected statistics to the optimizer.

Note: IS_REOPTIMIZABLE column shows whether the next execution matching this child cursor will trigger a re-optimization or if the child cursor contains re-optimization information, but will not trigger re-optimization because the cursor was compiled in reporting mode.

Quiz

A query plan changes during execution because runtime conditions indicate that optimizer estimates are inaccurate.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Lesson Road Map

- Adaptive SQL Plan Management
- Adaptive Execution Plans
- Optimizer Statistics Management



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You now move on to optimizer statistics management.

SQL Plan Directives

- A SQL plan directive is additional information and instructions that the optimizer can use to generate a better plan:
 - Collect missing statistics.
 - Create column group statistics.
 - Perform dynamic sampling.
- Directives can be used for multiple statements:
 - Directives are collected on query expressions.
- They are persisted to disk in the SYSAUX tablespace.
- Directives are automatically maintained:
 - Created as needed during compilation or execution:
 - Missing statistics, cardinality misestimates
 - Purged if not used after a year



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

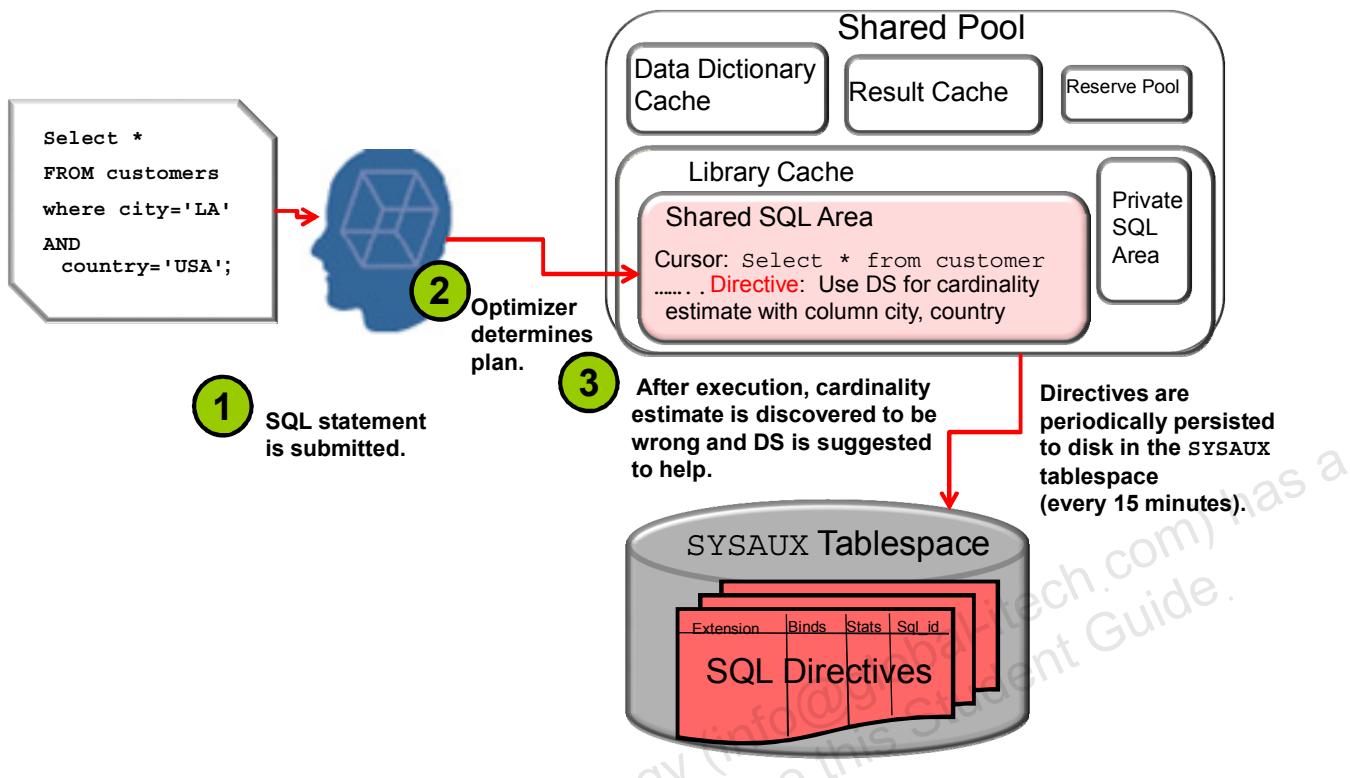
In previous releases, the database stored compilation and execution statistics in the shared SQL area, which is non-persistent. Starting in Oracle Database 12c, the database can use a SQL plan directive, which is additional information and instructions that the optimizer can use to generate a more optimal 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, 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.

Creating SQL Plan Directives

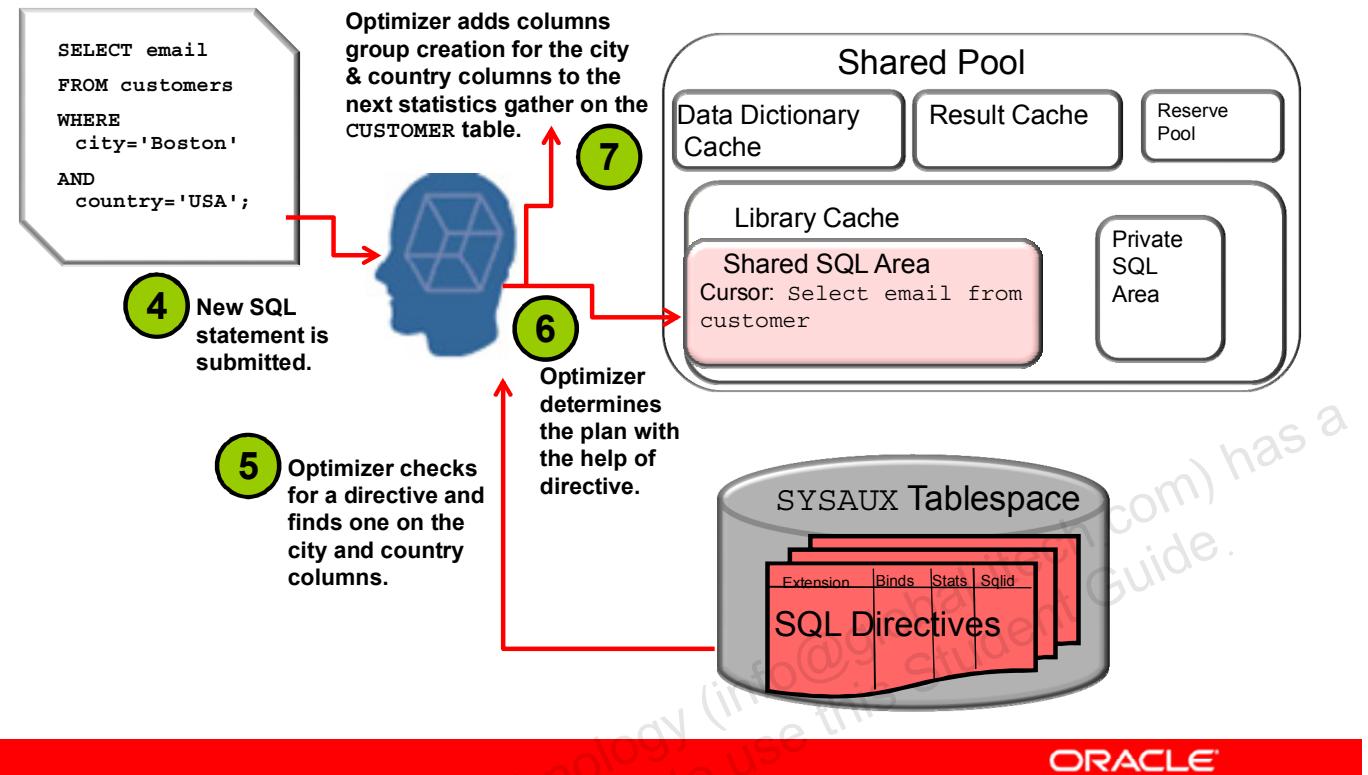


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

The slide shows how SQL Directives are created. The SQL statement is submitted, and then the optimizer determines the plan. After execution, when cardinality estimates are discovered to be wrong, dynamic sampling (DS) is suggested to help. Directives are periodically persisted to disk in the SYSAUX tablespace.

Using SQL Plan Directives



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When a new SQL statement is submitted, the optimizer checks for the directives and finds one on the city and country columns. After the optimizer gets the directive, it determines the plan with the help of the directive. The optimizer adds columns group creation for the city and country columns to the next statistics gather on the CUSTOMERS table.

SQL Plan Directives: Example

```
B.1. Manually Flush Directives (auto flush done every 15 minutes)
=====
SQL> exec dbms_spd.flush_sql_plan_directive
PL/SQL procedure successfully completed.

B.2. Display Directives
=====

SQL> col state format a5
SQL> col subobject_name format a11
SQL> col object_name format a13
SQL> select object_name, SUBOBJECT_NAME, type, state, reason from dba_sql_plan_directives d,
and object_name in ('ORDER_ITEMS', 'PRODUCT_INFORMATION');

OBJECT_NAME  SUBOBJECT_N TYPE      STATE REASON
-----
ORDER ITEMS   UNIT PRICE DYNAMIC SAMPLING NEW  SINGLE TABLE CARDINALITY MISESTIMATE
ORDER ITEMS   QUANTITY  DYNAMIC SAMPLING NEW  SINGLE TABLE CARDINALITY MISESTIMATE
ORDER ITEMS           DYNAMIC SAMPLING NEW  SINGLE TABLE CARDINALITY MISESTIMATE
```

B.3. Use Directives

```
=====
SQL> explain plan for
select /*Q10*/ product_name
from order_items o, product_information p
where o.unit_price = 15
and quantity > 1
and p.product_id = o.product_id;
Explained.

SQL> select * from table(dbms_xplan.display(format=>'basic +note'));
PLAN_TABLE_OUTPUT
-----
Plan hash value: 2615131494

| Id | Operation          | Name          |
|---|---|---|
| 0 | SELECT STATEMENT |             |
| 1 |  HASH JOIN        |             |
| 2 |    TABLE ACCESS FULL| ORDER_ITEMS |
| 3 |    TABLE ACCESS FULL| PRODUCT_INFORMATION |

Note
-----
- dynamic sampling used for this statement (level=2)
- 1 Sql Plan Directive used for this statement
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This slide shows SQL directives examples.

You can monitor SQL plan directives by using the DBA_SQL_PLAN_DIR_OBJECTS view.

The Notes section of the plan display functionality shows you information about SQL Plan Directives also, such as one SQL plan directive used for this statement.

Online Statistics Gathering for Bulk-Load

- In Oracle Database 12c, the database automatically gathers table statistics during the following types of bulk loads:
 - CREATE TABLE AS SELECT
 - INSERT INTO . . . SELECT into an empty table by using a direct path insert.
- Statistics are available immediately after load.
- No additional table scan is required to gather statistics.
- All internal maintenance operations that use CTAS (MV refresh) will benefit.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

With online statistics gathering, statistics are automatically created as part of a bulk load operation such as a CREATE TABLE AS SELECT operation or an INSERT INTO . . . SELECT operation on an empty table.

Online statistic gathering eliminates the necessity to manually gather statistics after a bulk data load has occurred. It behaves in a similar manner to the statistics gathering done during a CREATE INDEX or REBUILD INDEX command. All internal maintenance operations that use CREATE TABLE AS SELECT (CTAS) ((MV) Materialized View refresh) benefit. In a data warehouse, you frequently need to load a large amount of data into the database. For example, a sales record data warehouse might load sales data nightly where online statistics gathering is useful.

Oracle Database 12c now gathers the statistics automatically, and the principal benefits are improved performance and manageability of bulk load operations by eliminating user intervention to gather statistics after the load. And also by removing an additional full table scan required for separate statistics gathering operations.

While gathering online statistics, the database does not gather index statistics or histograms. If index statistics or histograms are required, then Oracle recommends running DBMS_STATS.GATHER_TABLE_STATS after the bulk load. By default, GATHER_TABLE_STATS only gathers missing statistics. Thus, if you invoke GATHER_TABLE_STATS after a bulk load, then the database only gathers index statistics or histograms, and not table and column statistics collected during the bulk load.

Concurrent Statistics Enhancements in Oracle Database 12c

- Auto statistics gather job uses concurrency.
- Prevent concurrent statistics gathering to make the system overwhelmed through careful resource usage capping.
- Allow gathering index statistics concurrently.
- Allow gathering of statistics for multiple partitioned tables concurrently.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Prior to Oracle Database 12c, the automatic statistics gather job gathered statistics one segment at a time. It now uses concurrency.

In Oracle Database 12c, you can run multiple statistics gathering tasks concurrently which may provide considerable improvements in statistics gathering time by increasing the resource utilization rate, in particular, on multi-CPU systems.

Employ smart scheduling mechanisms to have maximum degree of concurrency (to the extent that resources are available), for example, allow gathering of statistics for multiple partitioned tables concurrently.

Statistics for Global Temporary Tables

- In Oracle Database 12c, global temporary tables can have a different set of optimizer statistics for each session.
- GLOBAL_TEMP_TABLE_STATS preference of the DBMS_STATS package controls whether to gather session or shared statistics for global temporary tables.
- DBMS_STATS commits changes to session-specific global temporary tables, but not to transaction-specific global temporary tables.
- The cursor using the session-specific statistics is not shared by other sessions.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A global temporary table is a special table that stores intermediate session-private data for a specific duration. Prior to Oracle Database 12c, like for permanent tables, only one set of statistics was available for all occurrences of global temporary tables.

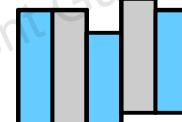
Starting with Oracle Database 12c, you can maintain session-private statistics for global temporary tables. You can choose to gather statistics for a global temporary table in one session and use the statistics only for this session. Meanwhile, you can continue to maintain a shared version of the statistics that is usable by all other sessions which have not yet gathered session-private statistics. Session level DML monitoring is also provided to the global temporary tables. Session-specific statistics for global temporary tables allow queries to be optimized more accurately based on your own data in a global temporary table.

Session-specific statistics have the following characteristics:

- Dictionary views that track statistics, such as DBA_TAB_STATISTICS, DBA_IND_STATISTICS, DBA_TAB_HISTOGRAMS, and so on have the SCOPE column to indicate whether the statistics are SHARED or SESSION-specific.
- Session-specific statistics reside only in memory and are never persisted to disk. The database automatically deletes the statistics when the session is exited.
- You can set the GLOBAL_TEMP_TABLE_STATS preference of the DBMS_STATS package to controls whether to gather session or shared statistics for global temporary tables. SESSION is the default.
- DBMS_STATS commits changes to session-specific global temporary tables, but not to transaction-specific global temporary tables so that data in transaction-specific global temporary tables is not lost.

Histogram Enhancements

- A histogram is a structure that represents the distribution of data for each column in a table.
- The optimizer uses histograms to compute the selectivity of filter and join predicates.
- In Oracle Database 12c, two new types of histograms are introduced:
 - Top Frequency histograms
 - Hybrid histograms
- With sampling set to AUTO, only frequency and hybrid histograms are created.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Histograms represent the data distribution in a table column. Oracle Database creates histograms on columns that have a data skew, to improve cardinality estimates.

In Oracle Database 12c, two new types of histograms have been introduced for columns, which have more than 254 distinct values to improve the cardinality estimates generated via histograms:

- Top Frequency histograms
- Hybrid histograms

These new histograms have a more accurate frequency of endpoint values, and enable the optimizer to choose better plans.

If Oracle Database 12c creates new histograms, and if the sampling percentage is AUTO, they are either frequency or hybrid, but not height-balanced.

Note: If you upgrade the database from Oracle Database 11g to 12c, any height-based histograms created before the upgrade remains in use.

Top Frequency Histograms

- Traditionally a frequency histogram is created only if NDV is less than the number of histogram buckets specified.
- Top frequency histogram created if:
 - Sampling percentage set to AUTO
 - Column contains more than nb distinct values
 - Top nb frequent values percentage is above $p\%$ of rows
- A Top Frequency histogram provides more accurate cardinality estimates.
- A Top Frequency histogram is indicated by the TOP-FREQUENCY value in the DBA_TAB_COL_STATISTICS.HISTOGRAM column.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Prior to Oracle Database 12c, a frequency histogram was created only if the number of distinct values (NDV) for the column was less than the number of histogram buckets specified (up to 254). With Oracle Database 12c, if a column contains more than 254 distinct values, and if a small number of distinct values occupy more than 99% of the rows, the database creates a Top Frequency histogram by using the small number of extremely popular distinct values. A Top Frequency histogram can produce a better histogram for highly popular values by ignoring statistically insignificant unpopular values.

A Top Frequency histogram is indicated by the TOP-FREQUENCY value in the DBA_TAB_COL_STATISTICS.HISTOGRAM column.

Starting in Oracle Database 12c, the database creates Top Frequency histograms when the following conditions are true:

- The sampling percentage is AUTO. In this case, the database constructs frequency histograms from a full table scan. For all other sampling percentage specifications, the database derives frequency histograms from a sample.
- The number of distinct values in the column is greater than the number of buckets (nb) created or defaulted.
- The percentage of rows occupied by the top nb frequent values is equal to or greater than threshold p , where p is $(1-(1/nb))*100$.

Note: Starting in Oracle Database 12c, the database constructs frequency histograms from a full table scan when the sampling size is AUTO (which is the default).

Hybrid Histograms

- A hybrid histogram combines characteristics of both height-based histograms and frequency histograms.
- It stores the endpoint repeat count value, which is the number of times the endpoint value is repeated, for each endpoint in the histogram.
- In Oracle Database 12c, the database creates hybrid histograms when the following conditions are true:
 - The sampling percentage is AUTO.
 - The criteria for top frequency histograms do not apply.
 - NDV is greater than nb , where nb is the user-specified number of buckets. If no number is specified, nb defaults to 254.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A hybrid histogram combines characteristics of both height-based histograms and frequency histograms.

The height-based histogram sometimes produces inaccurate estimates. For example, a value that occurs as an endpoint value of only one bucket but almost occupies two buckets is not considered as popular.

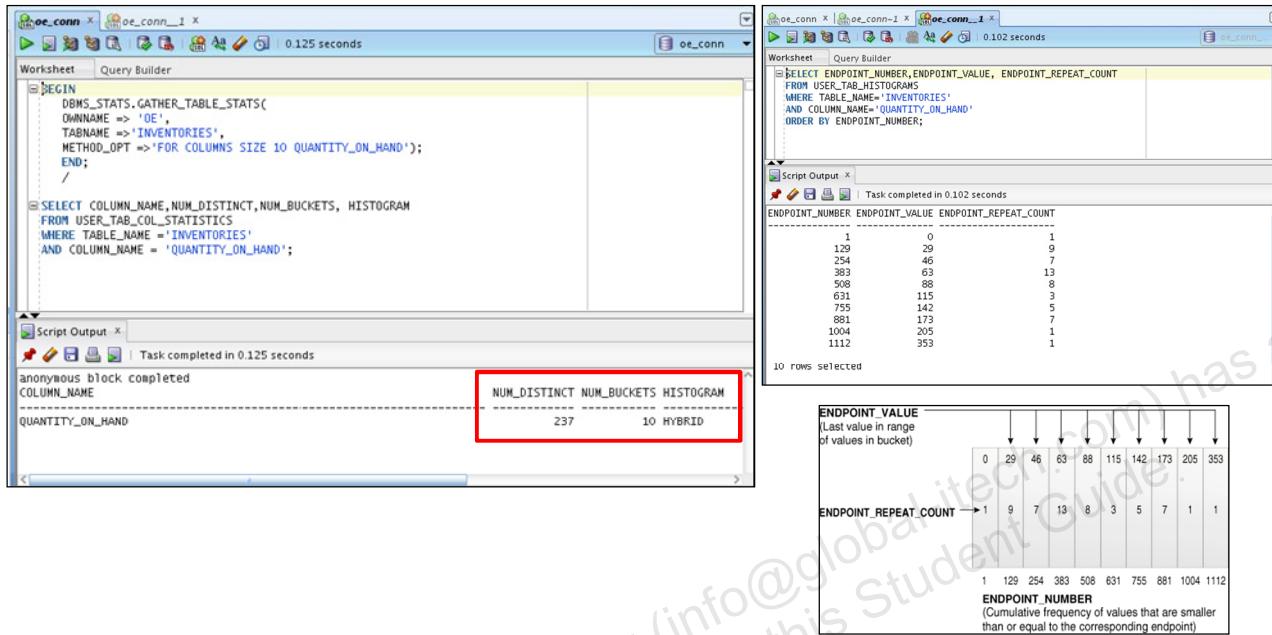
To solve this problem, a hybrid histogram stores the endpoint repeat count value and the cumulative frequency of values that are smaller than the corresponding endpoint value for each bucket in the histogram. By using these two pieces of information, the optimizer can obtain accurate estimates for almost popular values.

In Oracle Database 12c, the database creates hybrid histograms when the following conditions are true:

- The sampling percentage is AUTO.
- The criteria for top frequency histograms do not apply.
- NDV is greater than nb where nb is the user-specified number of buckets or the default of 254.

Hybrid Histograms: Example

You can view hybrid histograms by using the `USER_TAB_COL_HISTOGRAMS` table.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the slide example, the `oe.inventories.quantity_on_hand` column has 237 distinct values and 1112 rows. The histogram distributes the data into 10 buckets, each with roughly the same number of rows. In query output, one row corresponds to each bucket in the histogram. Oracle Database added a special 0 bucket to this histogram because the value in the first bucket (29) is not the minimum value for the `quantity_on_hand` column. The 0 bucket holds the minimum value of 0 for `quantity_on_hand`.

The graphic represents the hybrid histogram for `inventories.quantity_on_hand`. The x-axis shows the endpoint (bucket) numbers, which represent the cumulative frequency of the associated values. The diagram shows that 63 is the value at the right border of the fourth bucket, and that this value is repeated 13 times in the bucket.

Extended Statistics Enhancements

- DBMS_STATS enables you to identify and create extended statistics.
- There are two types of extended statistics:
 - Column group statistics
 - Expression statistics
- Extended statistics are automatically maintained when statistics are gathered on the table.
- Oracle Database 12c can detect and create the necessary column groups for you based on a workload.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Extended statistics allow statistics to be gathered on a group of columns within a table. DBMS_STATS enables you to identify and create extended statistics, which are statistics that can improve cardinality estimates when multiple predicates exist on different columns of a table.

There are two types of extended statistics:

- Column group statistics are useful when multiple columns from the same table are used in WHERE clause predicates.
- Expression statistics are useful when a column is used as part of a complex expression in WHERE clause predicates.

Starting in this release, Oracle Database can determine which column groups are required in a given workload or SQL tuning set (STS), and then can create the associated column groups. Thus, for any given workload, you no longer need to know which columns from each table will be used together. By monitoring a workload, Oracle Database 12c records the necessary column groups information.

Subsequent slides show you how to detect useful column groups for a specific workload.

Capturing Column Group Usage

Enable workload monitoring

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> -- Switch on seed column usage for 300 seconds
SQL> EXEC dbms_stats.seed_col_usage(null, null, 300)
PL/SQL procedure successfully completed.
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In an Oracle SQL*Plus session, connect as `SYS` and run the `PL/SQL` program shown in the slide to enable monitoring for 300 seconds (the first argument refers to `SQLSET_NAME` and the second to `OWNER_NAME`).

The new `SEED_COL_USAGE` procedure iterates over the SQL statements in the specified SQL Tuning Set (here, all the statements that will be executed in the system for the next 300 seconds), compiles them, and then seeds column usage information in the data dictionary for the columns that appear in these statements.

Capturing Column Group Usage: Running the Workload

```
SQL> EXPLAIN PLAN FOR
  2  SELECT *
  3  FROM   customers
  4  WHERE  cust_city='Los Angeles'
  5  AND    cust_state_province='CA'
  6  AND    country_id=52790;
```

Explained.

PLAN_TABLE_OUTPUT

Plan hash value: 2008213504

| Id Operation | Name | Rows |
|----------------------------------|------|------|
| 0 SELECT STATEMENT | | 1 |
| 1 TABLE ACCESS FULL CUSTOMERS | | 1 |

8 rows selected.

Actual number of rows returned by the 1st query is 932. Optimizer underestimates because it assumes each predicate will reduce the number of rows.

```
SQL> EXPLAIN PLAN FOR
  2  SELECT country_id, cust_state_province, count(cust_city)
  3  FROM   customers
  4  GROUP BY country_id, cust_state_province;
```

| Id Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
|--|------|-------|-------|-------------|----------|
| 0 SELECT STATEMENT | | 1949 | 31184 | 408 (1) | 00:00:01 |
| 1 HASH GROUP BY | | 1949 | 31184 | 408 (1) | 00:00:01 |
| 2 TABLE ACCESS STORAGE FULL CUSTOMERS | | 55500 | 867K | 406 (1) | 00:00:01 |

Actual number of rows returned by the 2nd query is 145. Optimizer overestimates because it assumes no relationship between country and state.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Following from the previous example, the database must observe a representative workload to determine the appropriate column groups. You do not need to run the queries themselves during the monitoring period. Instead, you can run EXPLAIN PLAN for some longer-running queries in your workload to ensure that the database is recording column group information for these queries.

The examples in the slide show the explain plans for two queries on the customer's table.

Reviewing Column Group Usage

```
SQL> SELECT dbms_stats.report_col_usage ('SH', 'CUSTOMERS')
2  FROM dual;
```

COLUMN USAGE REPORT FOR SH.CUSTOMERS

| | | | |
|---|------------|---|---|
| 1. COUNTRY_ID | : EQ | → | EQ means that the column was used in the equality predicate in query 1. |
| 2. CUST_CITY | : EQ | | |
| 3. CUST_STATE_PROVINCE | : EQ | | |
| 4. (CUST_CITY, CUST_STATE_PROVINCE, COUNTRY_ID) | : FILTER | ↑ | FILTER means that columns are used together as filter predicates rather than as joins and so on. It comes from query 1. |
| 5. (CUST_STATE_PROVINCE, COUNTRY_ID) | : GROUP_BY | | |

In query 2, the GROUP_BY columns are used in the GROUP_BY expression.

FILTER means that columns are used together as filter predicates rather than as joins and so on. It comes from query 1.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The new REPORT_COL_USAGE procedure reports the recorded column group usage information.

The example in the slide reviews the column group usage information recorded for the CUSTOMERS table.

All three columns appeared in the same WHERE clause, so the report shows them as a group filter.

In the second query, two columns appeared in the GROUP_BY clause, so the report labels them as GROUP_BY.

The sets of columns in the FILTER and GROUP_BY report are candidates for column groups.

Creating Column Groups Detected During Workload Monitoring

```
SQL> SELECT dbms_stats.create_extended_stats ('SH', 'CUSTOMERS')
  2  FROM   dual;
```

```
#####
EXTENSIONS FOR SH.CUSTOMERS
.....
1. (CUST_CITY, CUST_STATE_PROVINCE, COUNTRY_ID) :
   SYS_STUMZ$C3AIHLPBROI#SKA58H_N      created
2. (CUST_STATE_PROVINCE, COUNTRY_ID) :
   SYS_STU#S#WF25Z#QAHIE#MOFFMM_      created
#####
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide creates column groups for the CUSTOMERS table based on the usage information captured during the monitoring window.

The database created two column groups for CUSTOMERS: one column group for the filter predicate and one group for the group by operation.

Note: After being created, you should regather statistics on the CUSTOMERS table.

Automatic Dynamic Sampling

- Starting in Oracle Database 12c, the optimizer automatically decides if dynamic sampling is useful and what dynamic sampling level will be used for all SQL statements.
- Automatic dynamic sampling is enabled when the `OPTIMIZER_DYNAMIC_SAMPLING` initialization parameter is set to the value of 11.
- Increase the scope to non-parallel statements.
- Extend capabilities to include joins and group by
- Make results of dynamic sampling queries persistent in cache to eliminate overhead of repeated sampling.
- Additional statistics sources (for example, extent maps) to compensate for stale statistics on volatile tables



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Prior to Oracle Database 12c, dynamic sampling always occurred during optimization, and only when a table in the query had no statistics. Starting in Oracle Database 12c, the optimizer can use automatic dynamic sampling. In this case, the optimizer automatically decides whether dynamic sampling is useful and which dynamic sampling level to use for all SQL statements.

The primary factor in the decision is whether available statistics are sufficient to generate a good plan. If the statistics are insufficient, the optimizer uses dynamic sampling. The scope of the statistics gathered by dynamic sampling has been increased to include joins, group by and nonparallel statements.

The statistics gathered by dynamic sampling are now persistent and may be used by other queries to eliminate overhead of repeated sampling.

It is also helpful to compensate for stale statistics on volatile tables.

Automatic dynamic sampling is enabled when the `OPTIMIZER_DYNAMIC_SAMPLING` initialization parameter is set to the value of 11 (the default value being 2).

Quiz

In Oracle Database 12c, the database creates hybrid histograms if the sampling percentage is AUTO.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned about:

- Adaptive SQL Plan Management
- Enhancements to the SQL management base
- How to use a dynamic plan to improve query performance
- Re-optimization for adaptive execution plans
- How to use SQL plan directives to generate a better plan
- Statistics gathering performance improvements
- How to use new histograms
- Enhancements to extended statistics and how to detect useful column groups for a specific workload
- Enhancements to dynamic sampling



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice 15 Overview: Using Adaptive Execution Plans and Gathering Statistics Enhancements

These practices cover the following topics:

- Using dynamic plans and re-optimization
- Using concurrent statistic gathering

16

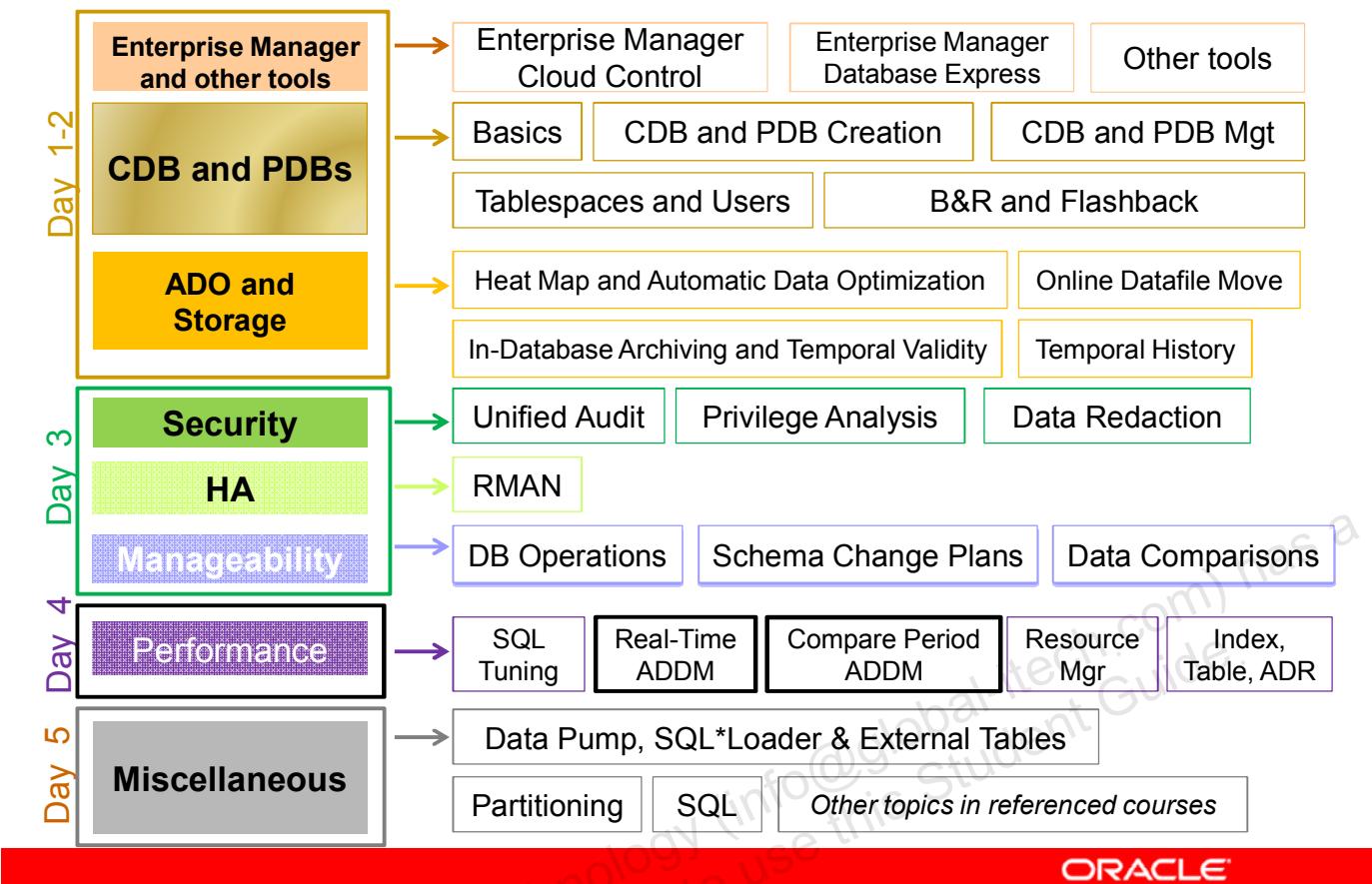
Emergency Monitoring, Real-Time ADDM, Compare Period ADDM, and ASH Analytics



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are new features about Automatic Database Diagnostic Monitor (ADDM): **Emergency Monitoring, Real-Time ADDM, Compare Period ADDM, and Active Session History (ASH) Analytics.**

Objectives

After completing this lesson, you should be able to:

- Explain Emergency Monitoring
- Describe Real-Time ADDM
- Use Real-Time ADDM
- Describe Compare Period ADDM
- Use Compare Period ADDM
- Generate a compare period ADDM report
- Identify the enhanced functionality used to view ASH data



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of the Real-Time ADDM, Emergency Monitoring, Compare Period ADDM and ASH Analytics features and usage, refer to the following sources of information:

- “*Oracle Enterprise Manager Cloud Control 12c: Database Management*” self-paced online course
- *Oracle Enterprise Manager Cloud Control 12c Demo Series* demonstrations:
 - Use Real-Time ADDM
 - Compare Period ADDM
 - Use Active Session History (ASH) Analytics

Emergency Monitoring: Challenges

- Sick systems
- Slow database
 - All users' queries are very slow.
 - Performance screens show slow data refresh rates.
 - There is a significant reduction in throughput .
- Database is hung due to internal contention for resources
 - Database is totally unresponsive; no logon is allowed.
 - Users' queries are hanging.
 - Performance screens do not refresh.



Solution: Shut the database instance down?

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The DBA detects that the system is sick based on different symptoms:

- Users complain that their queries do not respond.
- The refresh rate of the EM Performance Page is slower and slower.
- The throughput is abnormally degrading.

You expect to be able to perform a regular performance analysis with ADDM, but there might be serious limitations:

- On slowly performing systems, the snapshots may not be available for that time period.
- Depending on severity of the performance issue, it may not be advisable or even possible to take AWR snapshots.
- You may just not be able to connect to the database because it is hanging.

Is a database instance shutdown the only solution? There might be another solution that is less drastic than bouncing the database instance.

To perform this analysis, you need to be able to connect and have a quick, lightweight analysis in order to check who is blocking the database, why it is hanging, and this analysis should not require I/O nor global resources such as enqueues or latches.

Emergency Monitoring: Goals



Running Emergency Monitoring should be the final resort before bouncing the database.

1. Switching to Emergency Monitoring allows you to:
 - Connect to the instance in diagnostic mode
 - View the Emergency Performance page with data collected refreshed in real time
 - View ASH data and Hang Analysis table of top blocking and blocked sessions and deadlocks, refreshed in real time
2. Viewing Hang Analysis data helps you in:
 - Killing the root blocker responsible for hangs/deadlocks
 - Shutting down and starting up the instance



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Before shutting down the instance, you can launch the Emergency Monitoring. It allows you to have a quick look and perform a performance analysis even when you cannot log to the instance with a normal connection.

Emergency Monitoring allows DBAs with SYSDBA credentials to connect in diagnostic mode and have a quick, lightweight analysis to check who is blocking the database, why it is hanging. This type of connection does not require I/O nor global resources.

In Enterprise Manager 11g, “Memory Access Mode” is enabled and disabled explicitly with click of a button. This is done to start/stop the collector process, which reads performance data from SGA.

In the new approach there is no collector process. You do not have to switch between modes. When you navigate to Emergency Monitoring, the agent connects directly to the SGA and starts collecting SGA data bypassing SQL retrieval layer to get performance statistics. It will stop collecting after you return back to the regular performance monitoring.

It shows data refreshed in real time, and ASH data and Hang Analysis table of top blocking and blocked sessions, deadlocks, refreshed in real time.

It uses historical data obtained from ASH buffers to populate the performance pages.

ASH buffers generally contain history of wait data over the past 60 minutes, but this is not guaranteed. ASH buffers may not have enough history in the following cases:

- If there is too much active session activity then ASH buffer may get filled up earlier than 1 hour and flushed to the disk.
- If there is high resource contention, the process responsible for writing data into ASH buffers may get stuck and therefore result in loss of data.

It will, on some occasions, reduce the amount of history provided to the user from usual 30 minutes to something lesser.

When Emergency Monitoring shows the session that blocks other users, you can kill that session. Otherwise, you can either shut down the database instance or attempt a deeper analysis.

Real-Time ADDM: Challenges

- Sick systems
- Slow database
 - All users' queries are still very slow.
 - Performance screens still show slow data refresh rates.
 - There is still a significant reduction in throughput .
- Database is hung due to other contention for resources
 - Database might still be unresponsive; logon is allowed or is not allowed.
 - Users' queries are still waiting.
 - Performance screens do not refresh faster.
 - *You did not find any blocking session to kill.*
 - *Emergency Monitoring does not provide the root cause.*



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You did not find any blocking session and the database is still slow. Nevertheless you do not want to bounce the database instance before trying another analysis.

Real-Time ADDM offers you the possibility to perform a complementary analysis, a root-cause analysis that Emergency Monitoring does not provide.

Real-Time ADDM: Goals



1. Switch to Real-Time ADDM before bouncing the instance.
 - Starts collecting performance data from all database instances
 - Analyzes recent data for systems paralyzed because of severe contention on local or global resources
 - Provides holistic analysis for systems experiencing unusually high (though not severe) database activity
 - Detects findings for the recent activity (past 10 minutes)
 - Offers actionable recommendations
2. Use the recommendations to solve.
3. Return back to regular Performance Monitoring.

Note: Can be invoked for a RAC environment

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Real-Time ADDM works in a very similar fashion as ADDM to analyze performance.

Regular ADDM runs using AWR snapshots and provides findings and recommendations to let you know what needs to be changed to improve the performance.

Real-Time ADDM does not access the AWR snapshots but ASH recent activity from SGA data.

You connect in either modes with `SYSDBA` privilege, depending on the state of the instance:

- Diagnostic mode: If you cannot get a connection
- Normal mode: If you can still connect

Then the analysis starts, collecting recent performance data from all database instances from SGA, analyzing data for systems that are paralyzed because of severe contention on local or global resources.

It provides holistic analysis for systems experiencing unusually high (though not severe) database activity. After the analysis is complete, you can ask for the findings.

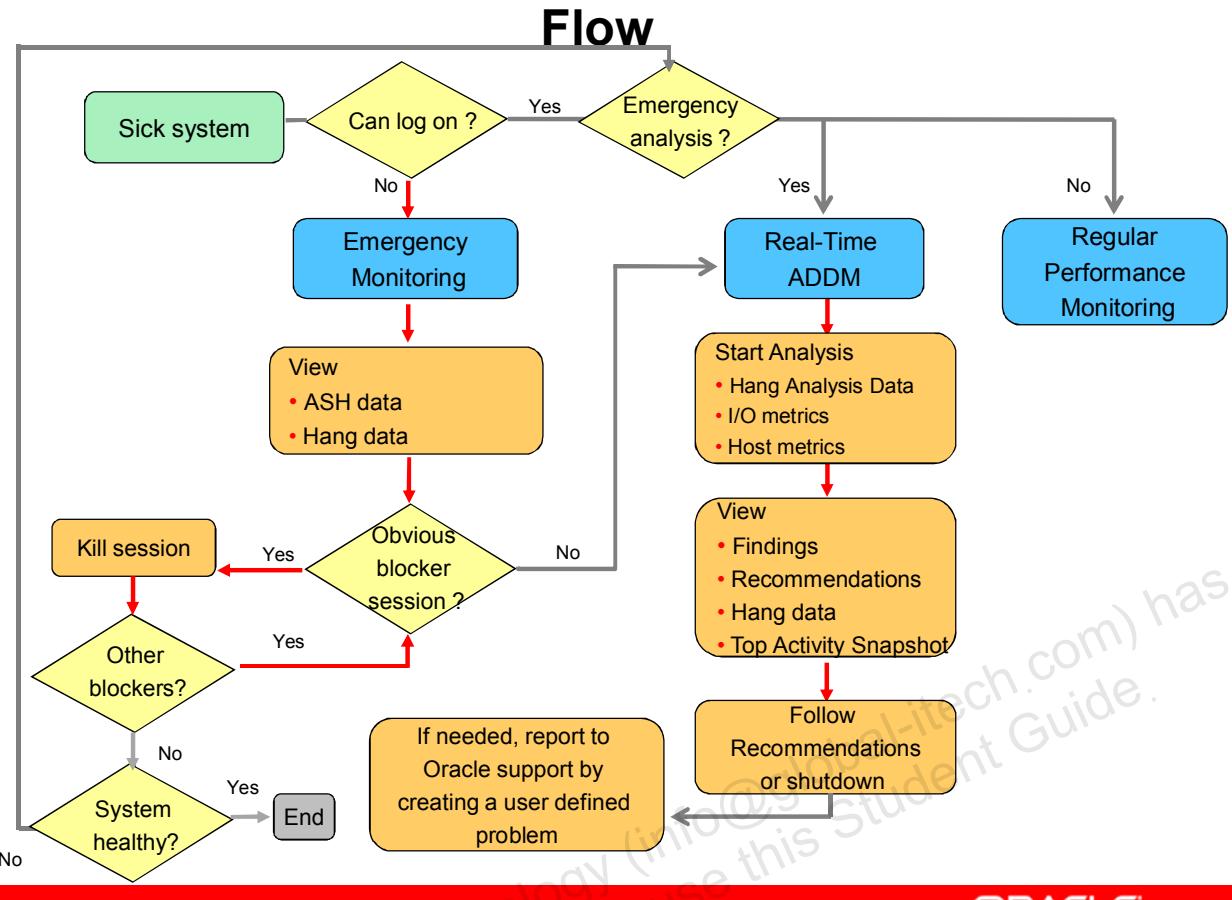
Real-Time ADDM finds that audit tracing cannot be performed due to lack of disk space and prevents connections, that a session needs to be killed because a logon trigger attempts to lock a table already locked, hence preventing any further connections. Real-Time ADDM identifies long-running queries that consume all resources.

In a case when a session blocks other users, the Hang Data tab presents the Hang Analysis page with the Final Blockers and the Blocked Sessions. You will see the details of the blocker session and thus be able to get the process ID to be killed.

The Top Activity Snapshot presents the top activity of the instance for the past 10 minutes. Because Real-Time ADDM focuses on the last recent period, it is useful only for clarifying current performance issues.

Always review the recommendations to get help on different solutions. Then return back to regular performance monitoring.

Real-Time ADDM can be invoked for a RAC environment in the same way as single instances.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can combine both Emergency Monitoring and Real-Time ADDM.

When you have to find a quick solution while your system is getting very slow or is hanging, use Emergency Monitoring to get a quick action to perform. Either find a blocking session to kill, or shut down the database instance and start it up.

If you do not want to shut down the database instance right away, perform Real-Time ADDM deeper analysis. It will collect data from SGA, perform an analysis to provide you a report of the root causes of the situation with recommendations and possible actions. Follow the recommendations if there are any. If not, then shut down your database instance.

In either cases, you can create a user-defined problem and use Support Workbench to report your issue to My Oracle Support.

Using the DBMS_ADDM Package

```
SQL> SELECT dbms_adm.real_time_adm_report()
  2  FROM dual;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can use new functions of the DBMS_ADDM package. The REAL_TIME_ADDM_REPORT returns a clob containing a Real-time ADDM report for the past five minutes.

Quiz

The difference between Real-Time ADDM and regular ADDM resides in the source used for analysis.

- a. Real-Time ADDM uses the AWR snapshots of the last 10 minutes.
- b. Real-Time ADDM uses the ASH recent activity from SGA data.
- c. Regular ADDM uses the AWR snapshots that are not yet purged.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

Real-Time ADDM works in a very similar fashion as regular ADDM to analyze performance; Real-Time ADDM does not access the AWR snapshots but ASH recent activity from SGA data.

AWR Compare Periods Report

Production performance inconsistency:

- Today, it is very bad.
 - Yesterday, it was excellent.
-
- What has changed? (unknown reason)
 - Why is there performance degradation?



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

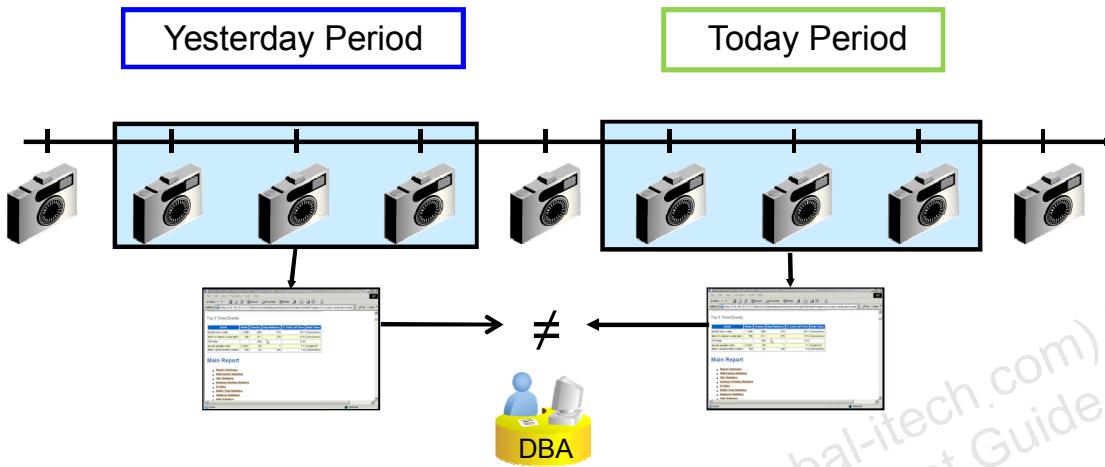
The challenge for DBAs about performance, when there is a degradation or improvement, is to identify what has changed and why, and which changes may have impacted the performance between two periods of time.

It can be normal if a different workload was executed, or if the storage devices changed, or if an ETL was executing while end users were executing their usual work, or simply the weekly ETL execution ran three more hours than last week.

DBAs need to compare comparable periods when analyzing performance changes.

Tool: Preserved Snapshot Sets

- Create a “yesterday” preserved snapshot set and a “today” preserved snapshot set.



- Run comparison between the two preserved snapshot sets.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Preserved Snapshot Sets

The diagram illustrates the Oracle AWR interface. On the left, the 'Automatic Workload Repository' page shows general settings like snapshot retention (7 days) and collection level (TYPICAL). It also lists 'Preserved Snapshot Sets' (labeled 1 and 2). A red box highlights 'Preserved Snapshot Set 2'. An arrow points from this box to the 'Preserved Snapshot Sets' table below. Another arrow points from the 'Preserved Snapshot Set 2' entry in the table to the 'Compare Periods: Review' screen on the right. This screen shows two periods: 'First Period' (Snapshot Set ID 1, AWR_1113404370198) and 'Second Period' (Snapshot Set ID 2, AWR_1113404430025). The 'Finish' button is highlighted with a red box.

| Select | Preserved Snapshot Set ID | Name | Beginning Snapshot ID | Beginning Snapshot Capture Time | Ending Snapshot ID | Ending Snapshot Capture Time |
|----------------------------------|---------------------------|-------------------|-----------------------|---------------------------------|--------------------|------------------------------|
| <input checked="" type="radio"/> | 1 | AWR_1113404370198 | 49 | Apr 12, 2005 5:50:48 AM | 50 | Apr 12, 2005 7:00:16 AM |
| <input type="radio"/> | 2 | AWR_1113404430025 | 74 | Apr 13, 2005 7:00:07 AM | 75 | Apr 13, 2005 8:00:11 AM |

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

When you created preserved snapshot sets, you were sure that the snapshots would not be purged from AWR after the retention period. You see here that we already have two identified preserved snapshot sets related to two different periods of time. The action of "Compare Periods" compared their respective AWR data sets.

AWR Comparison with AWR Snapshot Sets

| Database Instance: orcl > Automatic Workload Repository > Compare Periods: Results | | | | Logged in As SYS | | |
|--|----------------------------|---------------------------------|--------------------------|--|------------------------------|-------------------------------|
| Compare Periods: Results | | | | Change Periods | | |
| First Period | | Second Period | | | | |
| Beginning Snapshot ID | 21 | Beginning Snapshot Capture Time | Mar 30, 2005 6:00:12 AM | | | |
| Ending Snapshot ID | 27 | Ending Snapshot Capture Time | Mar 30, 2005 12:00:52 PM | | | |
| Second Period | | First Period | | | | |
| Beginning Snapshot ID | 45 | Beginning Snapshot Capture Time | Mar 31, 2005 6:00:10 AM | | | |
| Ending Snapshot ID | 51 | Ending Snapshot Capture Time | Mar 31, 2005 12:00:07 PM | | | |
| General | | Report | | | | |
| View Data | Per Second | | | Previous 1-27 of 27 Next | | |
| Name | First Period Metric Ratio | Second Period Metric Ratio | First Period Value | Second Period Value | First Period Rate Per Second | Second Period Rate Per Second |
| DB cpu (seconds) | | | 0.00 | 0.00 | 0.00 | 0.00 |
| DB time (seconds) | | | 63,816.00 | 63,804.81 | 2.95 | 2.95 |
| db block changes | | | 199,031.00 | 125,993.00 | 9.20 | 5.83 |
| execute count | | | 126,891.00 | 123,762.00 | 5.86 | 5.73 |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This slide shows a portion of the results of the AWR Compare Periods operation in Oracle Database 11g, which identifies statistical differences between two snapshot periods. For example, this report shows that there were significantly more “db block changes” in the first period.

Clicking the Report link on this page displays an HTML report comparing the two periods, showing differences in areas such as wait events, OS statistics, services, SQL statistics, instance activity, IO statistics, segment statistics, and so on.

AWR Compare Periods: Examples

Production performance inconsistency:

- The change is known.
 - Upgrade
 - Schema (indexing, partitioning table)
 - Parameter
 - Statistics recollection
 - OS platform
 - OS upgrades
 - CPU, memory
 - Storage (from FS to ASM)
- Why is there performance improvement or degradation?



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The DBA is in a situation where the performance has degraded or improved, and he knows the changes that have been completed. Tests were completed:

- Database upgrade
- Schema changes
- Parameter changes
- Use of new optimizer enhancements
- System or I/O statistic collection
- Use of a new type of storage such as ASM instead of file systems
- Memory increase or decrease to verify the impact
- More CPU added or new nodes in the RAC environment

In these situations, the DBA knows what has changed, but he does not know why the performance changed. Why does the change impact the performance system?

AWR Comparison with DB Replay

Run comparison between DB Replay capture and replay reports, or between two different replay reports.

The screenshot shows the Oracle Enterprise Manager interface with two main windows:

- Workload Profile:** Shows a summary of the workload profile named "replay1_jfv". It includes details like Capture Name: "capturejfv1", Duration: "00:02:34", and Start Time: "Jul 10, 2007 9:46:32 PM GMT+07:00".
- Workload Repository Compare Period Report:** This window contains two tabs: "Host Configuration Comparison" and "System Configuration Comparison".

A red box highlights the "Creating Report..." button in the center of the interface. A red arrow points from this button towards the "Host Configuration Comparison" tab in the adjacent window.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In this case, you use Real Application Testing with a DB Replay capture before changes and a DB Replay replay after changes. Then you run an AWR Compare Periods Report between the capture and the replay periods. Or you can change a factor and replays again the capture. Then you run an AWR Compare Periods Report between the first replay and the second replay. This type of report displays metrics values collected in the first and second period.

What is Missing?

Only AWR statistics reported in both cases:

- Comparison between two time periods
- Comparison between DB Replay capture and replay or two replays



Missing intelligent reporting with analysis:

- Changes that happened
- Mapping root causes to performance degradation effects



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The AWR Compare Periods report is interesting but you still have to perform an analysis on these metrics to find the effects that were mapped to the root causes of the performance degradation or improvement.

Analysis

| Causes Identify system changes | Effects Identify performance difference | Map effects to causes with a rule set |
|---|--|--|
| <ul style="list-style-type: none"> • Configuration changes <ul style="list-style-type: none"> – DB version • Workload changes <ul style="list-style-type: none"> – Change in SQLs – Database time consumed by these SQLs | <ul style="list-style-type: none"> • Computes problem impacts with ADDM methodology <ul style="list-style-type: none"> – In base period – In compare period • Measures the difference | <ul style="list-style-type: none"> • Rules triggered by performance changes <ul style="list-style-type: none"> – SGA_TARGET decrease can cause I/O increase |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Compare Period ADDM, as opposed to the former method, performs a cause-to-effect analysis.

1. It first identifies the system changes that may have caused the performance change. For example, it detects a configuration change in DB version, or a workload change with SQL changes. These causes can cause performance difference.
2. Then it identifies the effects of these particular changes. For that purpose, it runs an ADDM analysis for the base period and one for the compare period and then measures the differences between both periods.
3. Finally, it maps the effects to the causes with rule sets. For example, an SGA_TARGET increase can cause an I/O increase.

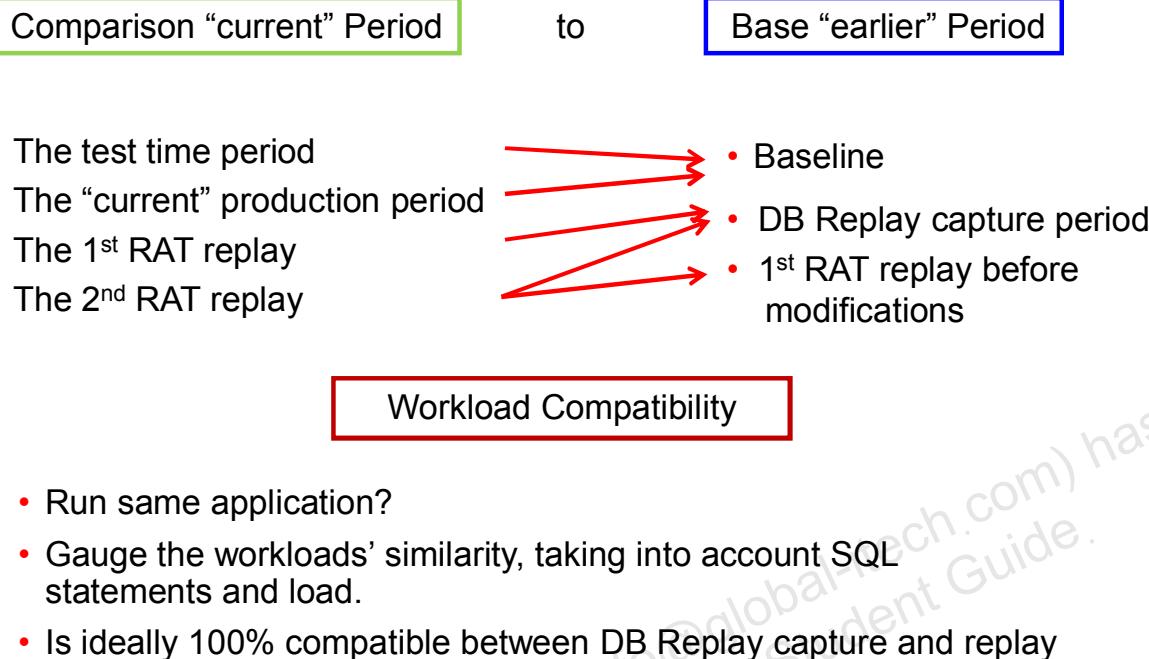
Compare Period ADDM provides more than what you had before with former methods, the identification of changes and an intelligent cause to effect analysis.

Is the change due to changes in the workload? Did some new application start running? Were there some changes at the OS level? Was an instance parameter changed?

The report displays changes in the resource demand at the hardware and software levels.

These and other questions are more easily answered using this advisor than with the previous AWR Compare Periods reports.

Workload Compatibility



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To generate a Compare Period ADDM report, you have to perform the following steps:

1. Define the Base Period as being the normal one
2. Define the Reference Period against which you compare the problematic period. The problematic period can cover situations like a period in a production database, a period when tests are performed in a test database, a period when using new optimizer enhancements, or another type of storage, or adding more memory, or more CPU or new nodes in a RAC environment.

In some cases, you use DB Replay, capturing and replaying where the Base Period would be the capture period, and the Compare Period the replay one.

A new factor comes into play during the comparison, the workload compatibility between two periods. Do you compare similar things? Are the SQL statements similar in both periods? Is it the same application running in both periods? This is called “SQL Commonality”.

This is a good way to gauge the workloads’ similarity, taking into account SQL statements and their respective load. In a live system, it is not 100% certain that DBAs compare exact or even similar application periods. If the result of the report shows an 80 or 90% compatibility level, then the DBA can rely on the findings provided by Compare Period ADDM.

Testing with Real Application Testing, you ideally get a 100% compatibility between a DB Replay capture and a replay or between two replays because the workload is packaged by DB Replay.

Comparison Modes

The screenshot shows the Oracle Enterprise Manager interface for a Container Database (CDB1.example.com). The 'Performance' tab is selected. The main area is titled 'Run Compare Period ADDM' with a sub-section 'Step 1: Select a Comparison Period'. It includes fields for 'Begin Time' (Aug 27, 2012 2:19:30 PM) and 'End Time' (Aug 27, 2012 3:19:46 PM), with a note that time will be adjusted to the capture time of the closest snapshot. Below this is 'Step 2: Select a Base Period' with three options: 'Offset' (selected), 'Baseline', and 'Customize'. Under 'Offset', 'Preceding Period' is chosen. Under 'Baseline', 'SYSTEM_MOVING_WINDOW' is selected. Under 'Customize', 'Begin Time' is set to Aug 26, 2012 2:19:56 PM and 'End Time' to Aug 26, 2012 3:20:09 PM, with a note that time will be adjusted to the capture time of the closest snapshot.

One snapshot offset

System moving window

Customized period

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

1. First, select the Comparison Period that you want to inspect where you noticed a performance degradation.
2. Compare it to the Base Period where the performance was acceptable and that reflects a similar workload.

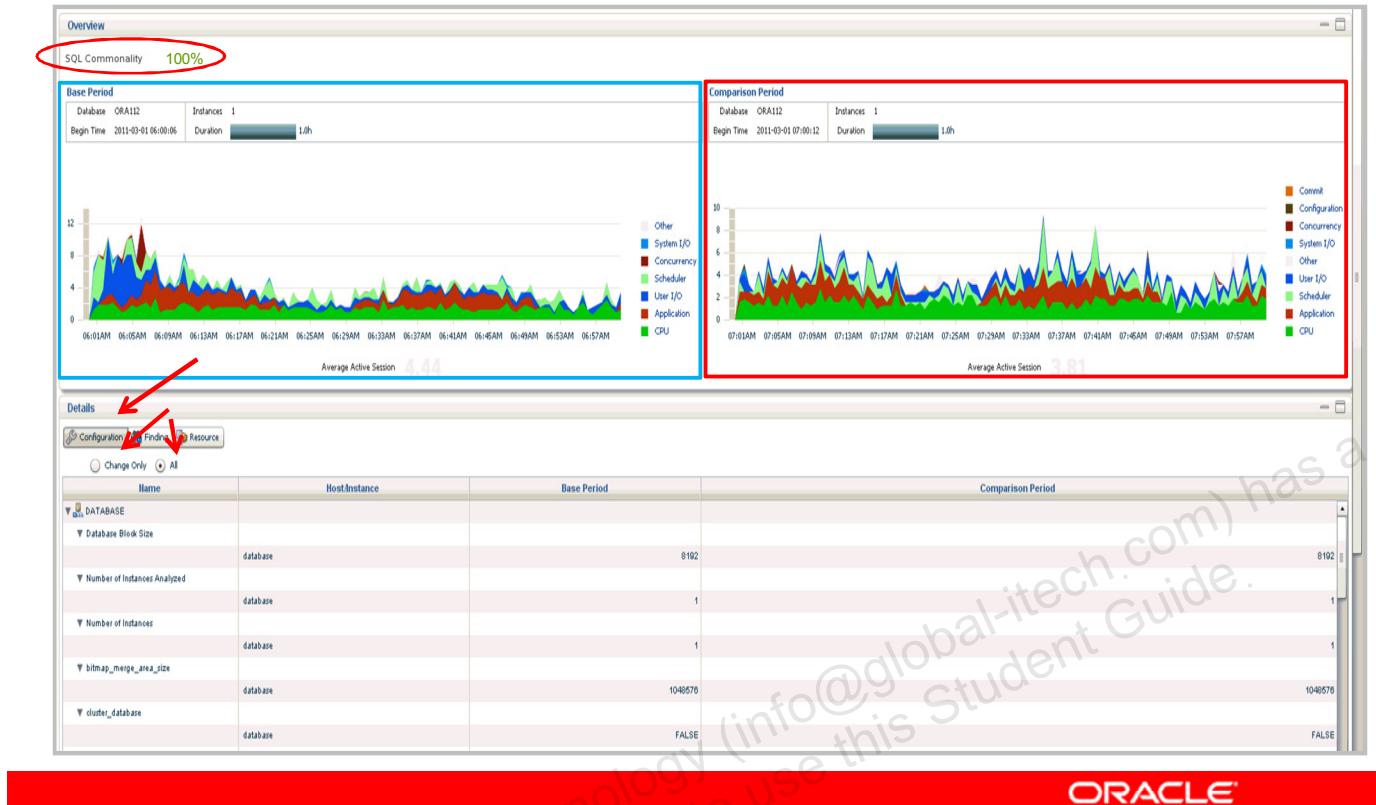
So you have three options to define the Base Period.

- The first option allows you to select an offset of one snapshot: in this example, the comparison period starts at 2 and ends at 3, therefore the base period will start at 1 and end at 2. In this same option, you can select an offset of one day, so it will use the previous day, same time as the comparison period. You can even choose a week back to compare with the performance of the previous week at the same time.
- The second option allows you to select a baseline, either the out-of-box `system_moving_window` baseline or any other baseline you created to reflect your normal application.
- The third option allows you to choose any other period of time.

In any case, it will automatically adjust the base period to a range of AWR snapshots that are as close as possible to the selected period of time.

3. And then run the report. It should come back with the details of the differences between both periods.

Report: Configuration



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The reporting shows various types of information, and not only a list of statistics as with AWR Compare Periods report. It displays graphics that are easier and faster to interpret.

The overview at the top of screen shows the two periods.

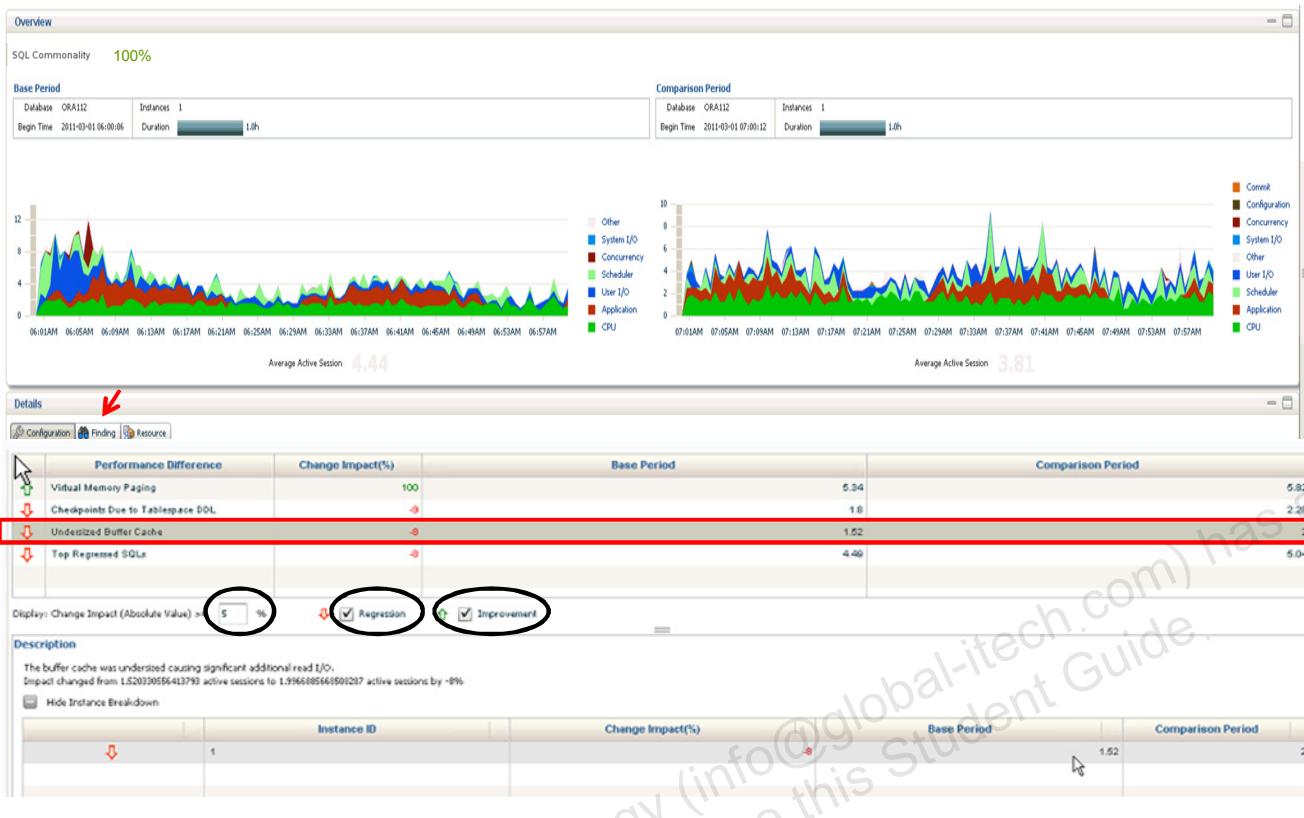
Between the two periods being compared, the report shows that they are 100% identical in terms of SQL compatibility. If the report shows a compatibility under 80%, you should not rely on this comparison because you compared two periods that are not comparable. So if you want a good, reliable comparison, check that the SQL commonality is at least 80 or 90%. Even if the compatibility is less than 80%, it is still worthwhile it to take a look at the findings. It will give an idea of what new SQLs were run in the system and how it affected the performance.

Now with this intelligent reporting, you get the changes that occurred between the two periods. In the details below the overview, you can choose to view only the configuration changes that occurred between the two periods, and not necessarily all the configuration information that is identical between the two periods.

If there was an instance parameter change, it will appear with the base period value and the comparison period value.

This information informs you about the existing changes but it does not explain why the performance degraded (or improved) and if this change had an impact on the performance degradation (or improvement). It is possible that this change did not impact the performance at all. It can be the contrary. It helped in performance. So another change might be the culprit.

Report: Finding



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You need to get the findings that explain what caused the performance degradation or improvement.

Here again you can choose to view only the areas where it degraded or improved, or all of them, and also set the percentage of impact for which you need to know the findings.

In the example in the slide, one of the reasons why the performance degraded is that the buffer cache size was undersized from 1.52 Mb to 2 Mb which was not sufficient. For each of the performance differences analyzed, you can get a detailed description of each finding and the SQL statements that were impacted positively or negatively.

Report: Resource CPU and I/O



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the Resource tab, the CPU page in the example on the slide shows a warning. There was, regarding the CPU consumption, more CPU consumed by Oracle and more Oracle Run Queue consumed also.

In the Resource tab, the I/O page shows the relative I/Os in both periods and tells you that neither periods were I/O Bound.

Report: Resource Memory

A screenshot of the Oracle Enterprise Manager interface. At the top, there are tabs: Configuration, Finding, and Resource. The Resource tab is highlighted with a blue border and has a red arrow pointing to it from above. Below the tabs, there are status indicators: CPU (yellow warning icon), Memory (yellow warning icon circled in red), I/O, and Interconnect. The Memory section shows a yellow warning icon. The main content area is titled 'Virtual Memory Paging' and compares two periods: 'Base' and 'Comparison'. The table shows memory usage for host 'host01.example.co'. The 'Total Physical Mem...' column shows 4,006.5MByte. The 'Total Memory Paging Out' column shows 56.6MByte for the base period and 50.3MByte for the comparison period. The 'Memory Used by Oracle Database' column shows 898.1Mbyte for the base period and 899.2Mbyte for the comparison period.

| Host Name | Total Physical Mem... | Total Memory Paging Out | | Memory Used by Oracle Database | |
|-------------------|-----------------------|-------------------------|-------------------|--------------------------------|-------------------|
| | | Base Period | Comparison Period | Base Period | Comparison Period |
| host01.example.co | 4,006.5MByte | 56.6MByte | 50.3MByte | 898.1Mbyte | 899.2Mbyte |
| | | | | | |
| | | | | | |

The Oracle logo, which consists of the word "ORACLE" in white capital letters on a red horizontal bar.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the Resource tab, the Memory page shows a warning. There is some swapping in both periods, but the values are very similar.

Using the DBMS_ADDM Package

Compare two periods within the same instance:

Comparison “current” Period
Snap_ID 123 to 124

to

Base “earlier” Period
Snap_ID 121 to 122

```
SQL> SELECT dbms_adm.compare_instances (
  2      base_dbid          => 1319927350,
  3      base_instance_id   => 1,
  4      base_begin_snap_id => 121,
  5      base_end_snap_id  => 122,
  6      comp_dbid          => 1319927350,
  7      comp_instance_id   => 1,
  8      comp_begin_snap_id => 123,
  9      comp_end_snap_id  => 124,
 10     report_type        => 'XML')
11  FROM dual;
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can use new functions of the DBMS_ADDM package. The DBMS_ADDM.COMPARE_INSTANCES returns a clob containing a compare period ADDM report comparing the performance of a single instance over two different time periods.

The parameters used are the following:

- BASE_DBID: DB ID of the base period
- BASE_INSTANCE_ID: Instance ID of the base period
- BASE_BEGIN_SNAP_ID: Snapshot ID of the beginning of the base period
- BASE_END_SNAP_ID: Snapshot ID of the end of the base period
- COMP_DBID: DB ID of the comparison period
- COMP_INSTANCE_ID: Instance ID of the comparison period
- COMP_BEGIN_SNAP_ID: Snapshot ID of the beginning of the comparison period
- COMP_END_SNAP_ID: Snapshot ID of the end of the comparison period
- REPORT_TYPE: Output type for the report—XML or HTML (Default is HTML)

In the slide, you see an ADDM comparison between two periods of time, each delimited in time by a begin and end snapshot.

Using the DBMS_ADDM Package

Functions to compare periods:

- COMPARE_INSTANCES
- COMPARE_DATABASES
- COMPARE_CAPTURE_REPLAY_REPORT
- COMPARE_REPLY_REPLY_REPORT



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Additional PL/SQL functions:

- COMPARE_INSTANCES: Generates a Compare Period ADDM report for an instance-level performance comparison.
- COMPARE_DATABASES: Generates a Compare Period ADDM report for a database-wide performance comparison.
- COMPARE_CAPTURE_REPLAY_REPORT: Generates a Compare Period ADDM report to compare performance for a Workload Capture to a Workload Replay.
- COMPARE_REPLY_REPLY_REPORT: Generates a Compare Period ADDM report to compare performance for one Workload Replay to another Workload Replay.

All the preceding functions return a clob.

Note: For a complete description of the available procedures, see the *Oracle Database 12c PL/SQL References and Types* documentation.

Quiz

The difference between an Compare Period ADDM report and an AWR Compare Periods report resides in the output format.

- a. True
- b. False

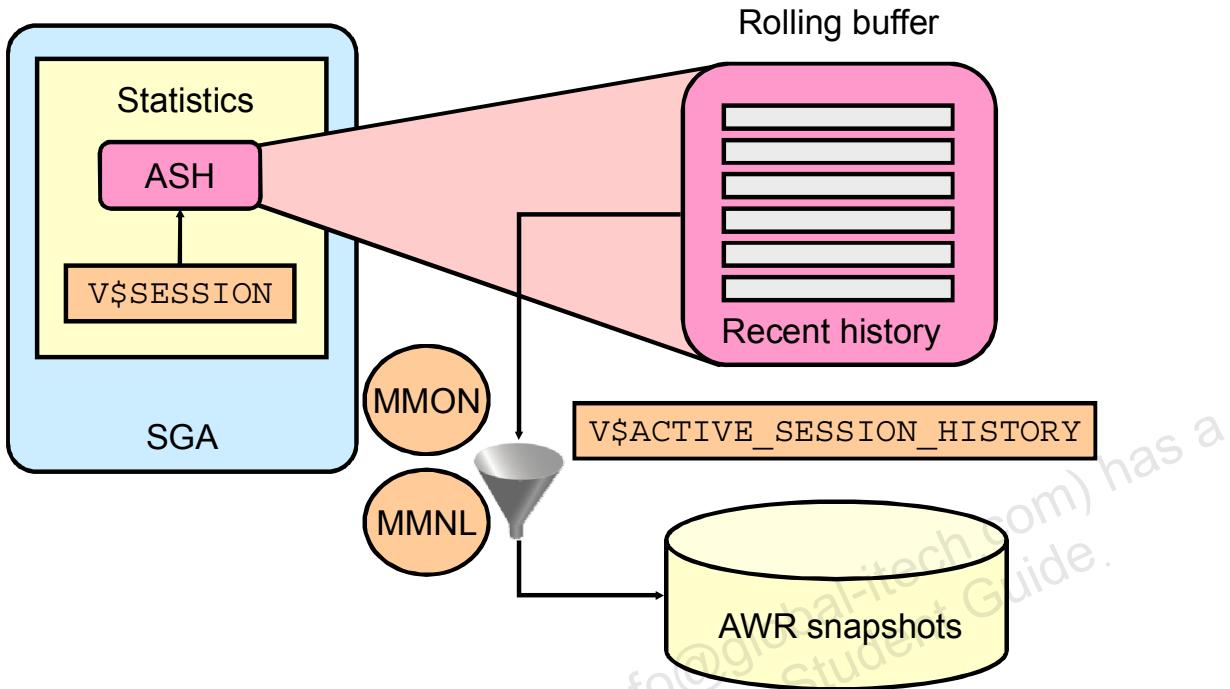


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

The Compare Period ADDM, compared to the AWR Compare Periods report, performs a cause-to-effect analysis.

ASH: Overview



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

One of the problems with AWR data is that, by its very nature, analysis of what is currently happening in a system requires detailed information about activity in the last five to 10 minutes. Because the AWR takes snapshots of the system every 60 minutes, the last snapshot could be almost an hour old. Therefore, the AWR does not contain enough information to perform current analysis.

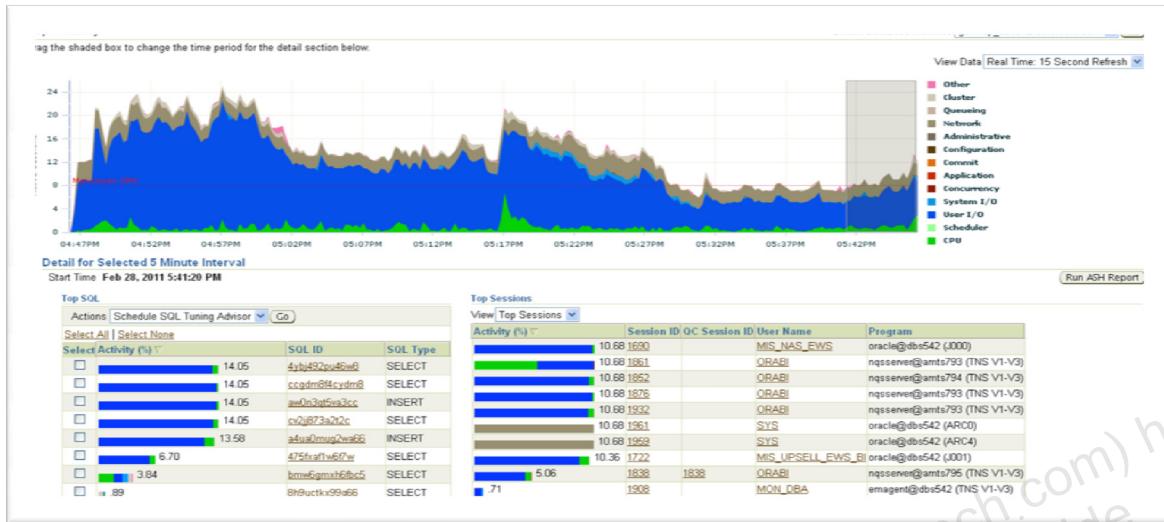
The Active Session History contains the history of recent session activity. Because recording session activity is expensive, ASH samples V\$SESSION every second and records the events that sessions are waiting for. Inactive sessions are not sampled. The sampling facility is very efficient because it directly accesses the internal database structures.

ASH is designed as a rolling buffer in memory; earlier information is overwritten when needed. The ASH statistics are available through the V\$ACTIVE_SESSION_HISTORY view. This view contains one row for each active session per sample.

Flushing all ASH data to disk is unacceptable because of its volume. The approach is to filter the data while flushing it to disk. This is done automatically by MMON every 60 minutes and by Manageability Monitor Light (MMNL) whenever the buffer is full.

The ASH memory comes from the SGA and is fixed for the lifetime of the instance. It represents 2 MB of memory per CPU. However, the size of ASH cannot exceed 5% of the shared pool size.

Top Activity Page



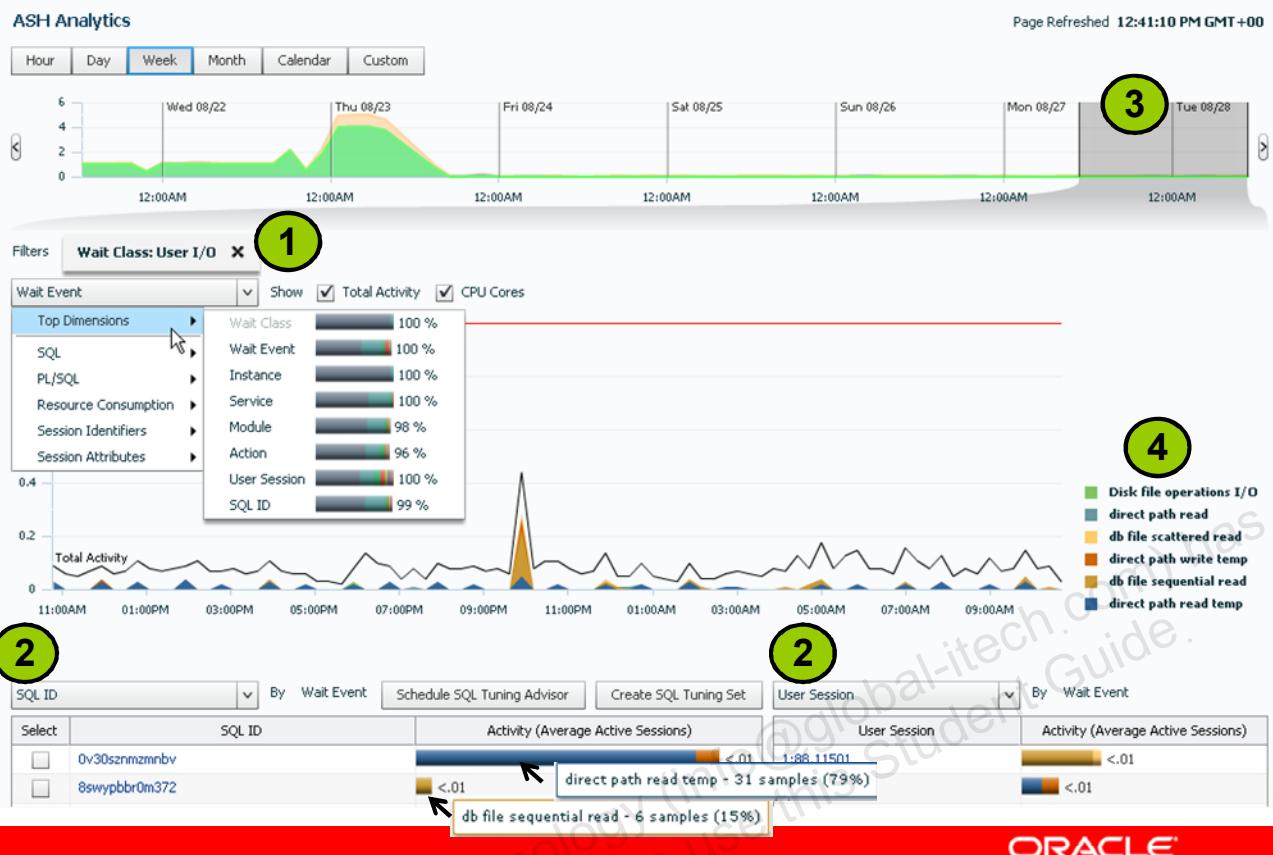
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Enterprise Manager 11g, the Top Activity page displays a lot of the ASH information. An example of this page is shown in the slide. There are a number of limitations in the way this information is displayed. These include:

1. You cannot switch dimensions on the area chart shown in the top part of the screen.
2. The left-hand table below that is fixed to displaying Top SQL, while the right-hand table has only a few dimensions that it can be displayed by, such as Top Sessions or Top Modules.
3. The information does not harness the full value of ASH data as some key dimensions that are actually captured with the ASH data are not displayed at all.
4. The slider used to select the time period for the detailed section is a fixed width (5 minutes real time, 30 minutes historical).
5. The visualization is restricted to a stacked area chart by wait classes so you always see the wait classes stacked over one another.
6. The data cannot be displayed as an active report, that is, it cannot be sent offline to other users for review.
7. There are limited drilldown capabilities that simply send you to other pages.
8. ASH reports are currently in text format and are not interactive.

ASH Analytics Page: Activity



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The new ASH Analytics page overcomes all these limitations allowing:

1. Filter dimensions on the Filters shown in the middle left part of the page
2. Select the dimensions for the left and right hand tables in the bottom left and right parts of the page
3. Vary the slider width to select the time period for the detailed section
4. Drill into a load map view to display the different waits indicating the importance of each by the size of the box

Summary

In this lesson, you should have learned how to:

- Describe Real-Time ADDM
- Use Real-Time ADDM
- Explain Emergency Monitoring
- Describe Compare Period ADDM
- Use Compare Period ADDM
- Generate a compare periods ADDM reports
- Identify the enhanced functionality used to view ASH data



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice 16 Overview: Emergency Monitoring and Compare Period ADDM

These practices cover the following topics:

- Using Emergency Monitoring
- Using Compare Period ADDM



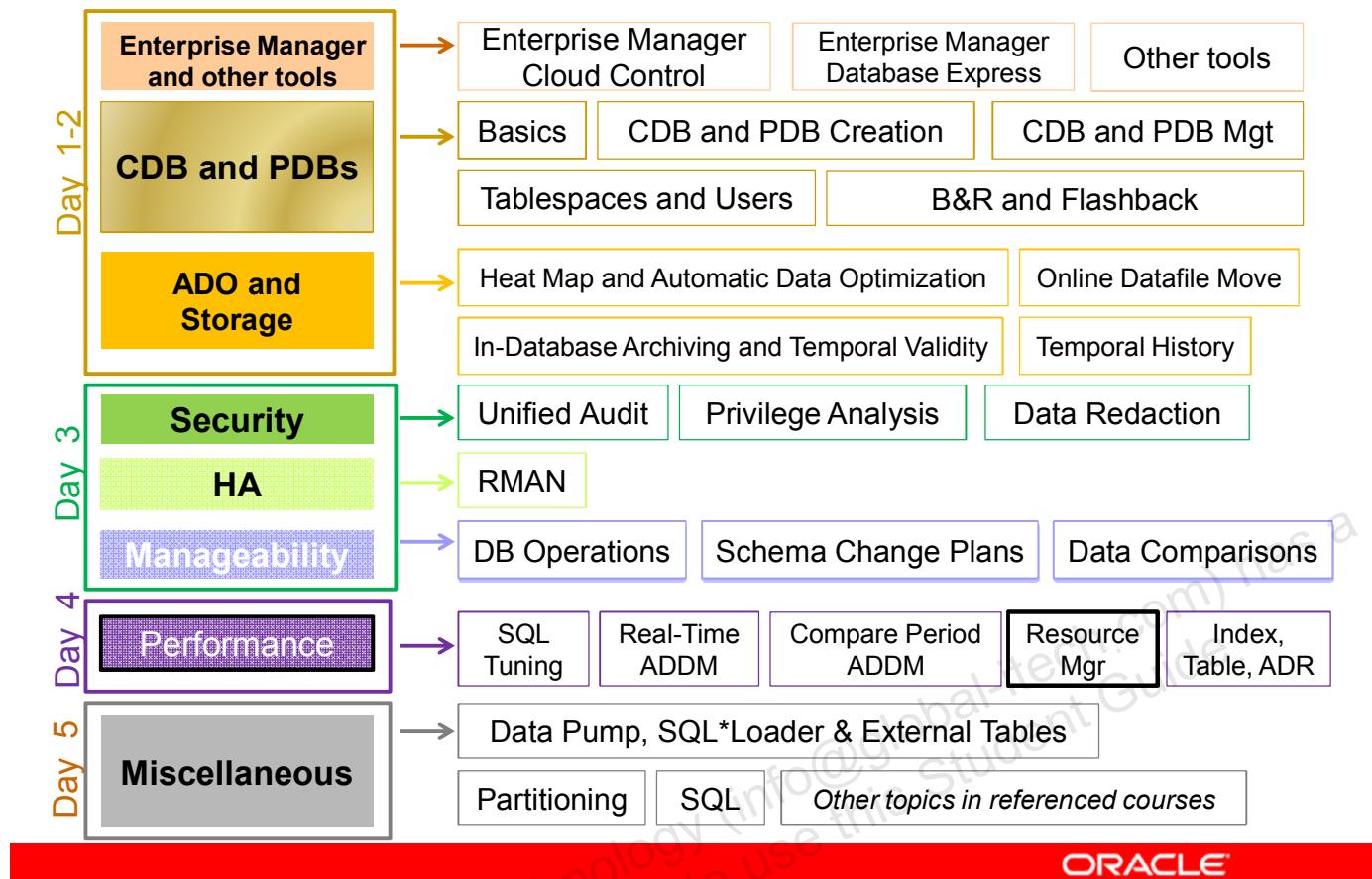
11

Resource Manager and Other Performance Enhancements

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several Resource Manager enhancements to handle resources within CDBs and PDBs. There is also a new multi-process multi-threaded Oracle architecture and enhancements on Database Smart Flash Cache and undo segments.

Objectives

After completing this lesson, you should be able to:

- Use Resource Manager in a CDB environment
- Describe the multi-process multi-threaded Oracle architecture
- Describe Database Smart Flash Cache enhancements
- Use temporary undo for your temporary tables



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of Oracle Pluggable Database new feature and usage, refer to the following guides in the Oracle documentation:

Oracle Database Administrator's Guide 12c Release 1 (12.1)

- Chapter “Using Oracle Resource Manager for Pluggable Databases with SQL*Plus”
- Chapter “Using Oracle Resource Manager for Pluggable Databases with Cloud Control”
- Chapter “Managing Resources with Oracle Database Resource Manager Runaway Queries”

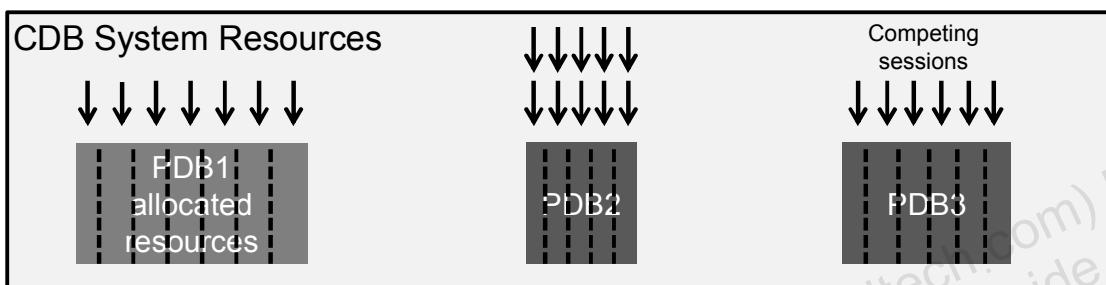
Refer to other sources of information:

- *Oracle Database 12c New Features Demo Series* demonstrations under Oracle Learning Library:
 - *Multitenant Architecture*
 - *Basics of CDB and PDB Architecture*

Resource Manager and Pluggable Databases

In a CDB, Resource Manager manages resources:

- Between PDBs
- Within each PDB



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In a non-CDB, you can use Resource Manager to manage multiple workloads that are contending for system and database resources. However, in a CDB, you can have multiple workloads within multiple PDBs competing for system and CDB resources.

In a CDB, Resource Manager can manage resources on two basic levels:

- **CDB level:** Resource Manager can manage the workloads for multiple PDBs that are contending for system and CDB resources. You can specify how resources are allocated to PDBs, and you can limit the resource utilization of specific PDBs.
- **PDB level:** Resource Manager can manage the workloads within each PDB.

Resource Manager allocates the resources in two steps:

1. It allocates a portion of the system's resources to each PDB.
2. In a specific PDB, it allocates a portion of system resources obtained in Step 1 to each session connected to the PDB.

Managing Resources Between PDBs

- PDBs compete for resources: CPU, Exadata I/O, parallel servers
 - System shares are used to allocate resources for each PDB.
 - Limits are used to cap resource utilization of each PDB.
- When a new PDB is plugged in, the CDB DBA can specify a default or explicit allocation



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

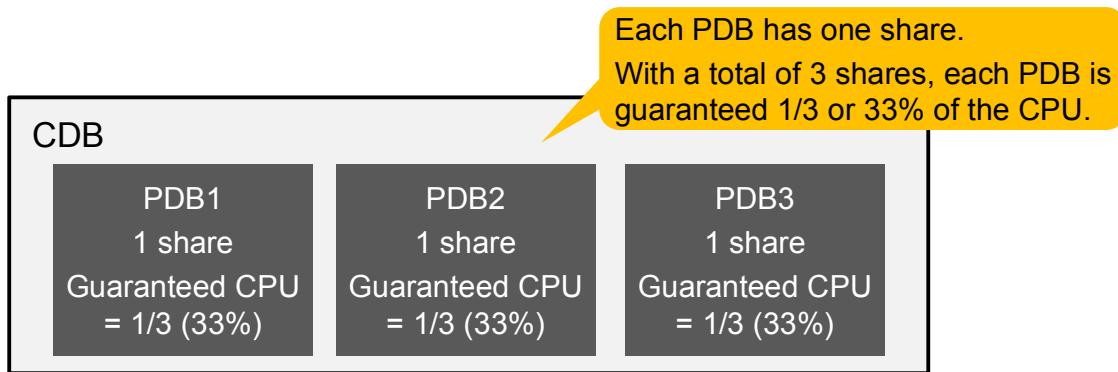
In a CDB with multiple PDBs, some PDBs typically are more important than others. The Resource Manager enables you to prioritize the resource (CPU and I/O, as well as allocation of parallel execution slaves in the context of parallel statement queuing) usage of specific PDBs. This is done by granting 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.

When a PDB is plugged in to a CDB and no directive is defined for it, the PDB uses the default directive for PDBs. As the CDB DBA, you can control this default and you can also create a specific directive for each new PDB.

Note: No consumer groups nor shares can be defined for the root container.

CDB Resource Plan Basics: Share



- Specify resources using “shares”.
 - Recomputation is not required when PDBs are added or removed.
 - PDB1 is guaranteed 33% of the CPU.
 - PDB1 is limited to 100% of the CPU.
 - PDB1 is actually using 20% of the CPU.

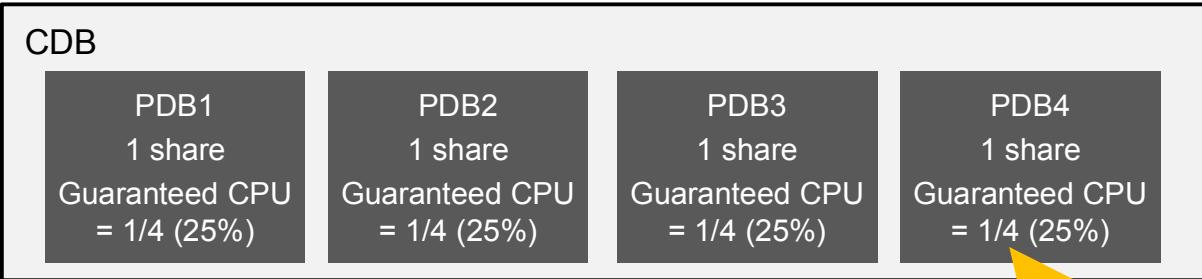
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To allocate resources among PDBs, you assign a share value to each PDB. A higher share value results in more resources for a PDB.

In this example, each existing PDB is assigned one share. With a total of three shares, each PDB is guaranteed to get at least 33% of the system resources.

Depending on the workload, each PDB can get up to 100% of the system resources but may actually use only 20% of these resources.

CDB Resource Plan Basics: Share



- Default allocation: one share

New PDB gets the default allocation:
1 share.
With a total of 4 shares, each PDB is
guaranteed 1/4 or 25% of the CPU.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

If a new PDB is added, it automatically gets the default share value of one.

In this case, with a total of four shares, each PDB is guaranteed to get 25% of the system resources.

Note: Each CDB resource plan gets a default directive added to it. You can change this default directive if its default value is not suitable for your plan.

CDB Resource Plan Basics: Share

CDB

| PDB1 | PDB2 | PDB3 | PDB4 |
|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 1 share | 1 share | 1 share | 2 shares |
| Guaranteed CPU = 1/5 (20%) | Guaranteed CPU = 1/5 (20%) | Guaranteed CPU = 1/5 (20%) | Guaranteed CPU = 2/5 (40%) |

If this PDB is more important, you can explicitly allocate it more shares. It then gets more resources than the other PDBs.

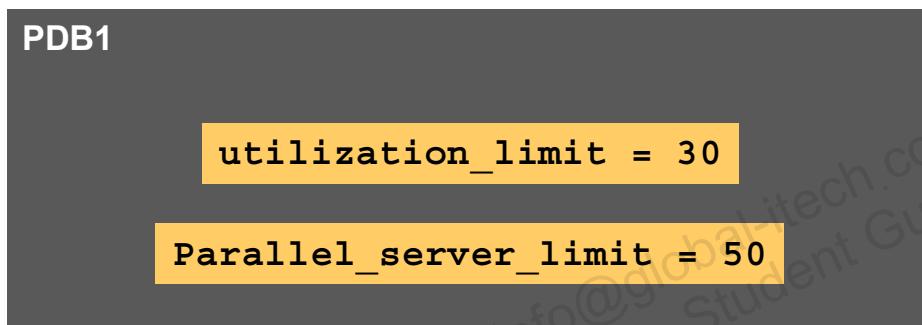
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

However, if you consider PDB4 to be of higher priority than the others, you can explicitly assign it more shares. In this example, PDB4 gets two shares while the other PDBs get one share. With a total of five shares, PDB1, PDB2, and PDB3 are guaranteed to get 1/5 (20%) of the system resources because they were assigned 1 share each. Because PDB4 was assigned 2 shares, it is guaranteed to get 2/5 (40%) of the system resources.

CDB Resource Plan Basics: Limits

- Two limits can be defined for each PDB:
 - Utilization limit for CPU, Exadata I/Os and parallel servers
 - Parallel server limit to override the utilization limit
- Default values: 100%
- You can change default values.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A limit restrains the system resource usage of a specific PDB. You can specify limits for CPU, Exadata I/Os, and parallel execution servers.

To limit the CPU, Exadata I/Os, and parallel servers (PARALLEL_SERVER_TARGET initialization parameter) usage of each PDB, you can create a plan directive using the UTILIZATION_LIMIT parameter expressed as a percentage of the system resources the PDB can use. Resource Manager throttles the PDB sessions so that the CPU, Exadata I/Os, and parallel servers utilization for the PDB does not exceed the utilization limit. In the slide example, PDB1 gets a maximum of 30% of the system CPU, Exadata I/Os bandwidth, and available parallel servers for the CDB instance.

For parallel execution servers, you can override the value defined by the UTILIZATION_LIMIT by creating a plan directive that uses the PARALLEL_SERVER_LIMIT parameter. The PARALLEL_SERVER_LIMIT value corresponds to a percentage of PARALLEL_SERVERS_TARGET. For example, if the PARALLEL_SERVERS_TARGET initialization parameter is set to 200 and the PARALLEL_SERVER_LIMIT is set to 50% for a PDB, then the maximum number of parallel servers the PDB can use is 100 (200 X .50).

When you do not explicitly define limits for a PDB, the PDB uses the default limits for PDBs that are set to 100%. These are the values corresponding to the default directive that is automatically added to your plan.

Note: You can also change the default directive attribute values for PDBs by using the UPDATE_CDB_DEFAULT_DIRECTIVE procedure in the DBMS_RESOURCE_MANAGER package.

CDB Resource Plan: Full Example

| PDB/Directive Name | Shares | Utilization Limit | Parallel Server Limit |
|-----------------------|--------|-------------------|-----------------------|
| (Default Allocation) | (1) | (100%) | (100%) |
| (Autotask Allocation) | (-1) | (90%) | (100%) |
| PDB1 | 1 | 30% | 50% |
| PDB2 | 1 | 30% | 80% |
| PDB3 | 1 | 30% | 30% |
| PDB4 | 2 | 100% | 100% |

PDB1 is

- Guaranteed 1/5 (20%) of CPU and Exadata disk bandwidth
- Limited to 30% of CPU and Exadata disk bandwidth
- Limited to 50% of the parallel servers

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This example shows you a possible CDB resource plan created in the root container.

The default allocation directive applies to PDBs for which specific directives have not been defined.

Default allocation is one share and both corresponding limits are set to their default of 100%. These are the default values for the default allocation directive.

The autotask allocation directive applies to automatic maintenance tasks that are run in the root or in PDBs.

The autotask allocation is -1 share, which means that the automated maintenance tasks get an allocation of 20% of the system resources. You should always change this default value. By default, the autotask allocation gets a utilization limit of 90% and 100% for its parallel server limit.

PDB1 is allocated 1 share which represents 1/5 or 20% of the CPU, Exadata disk bandwidth, and queued parallel queries will be selected 1/5 of the time compared to the other PDBs. In addition, a utilization limit of 30% of the resources is set as well as a 50% limit of the available parallel servers.

Root

Creating a CDB Resource Plan

1. Create a pending area.
2. Create a CDB resource plan.
3. Create directives for the PDBs.
4. (*Optional*) Update the default PDB directives.
5. (*Optional*) Update the default autotask directives.
6. Validate the pending area.
7. Submit the pending area.

DBMS_RESOURCE_MANAGER

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

CDB resource plan management is done directly from the root container.

The slide shows you the different steps you can use to create a CDB resource plan.

You use the DBMS_RESOURCE_MANAGER package to create a CDB resource plan and define the directives for the plan.

Note: If you try to define a CDB resource plan in a PDB, you get an error.

Creating CDB Resource Plan: SQL Example

```
SQL> CONNECT sys AS SYSDBA
```

```
SQL> EXEC DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
```

```
SQL> BEGIN
 2   DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN(
 3     plan      => 'daytime_plan',
 4     comment  => 'CDB resource plan for mycdb');
 5 END;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

From the root container of your CDB, connect as the `SYS` user.

Then, create a pending area using the `CREATE_PENDING_AREA` procedure.

You can now create a CDB resource plan named `daytime_plan` using the `CREATE_CDB_PLAN` procedure.

Creating CDB Resource Plan: SQL Example

```
BEGIN  
    DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE(  
        plan          => 'daytime_plan',  
        pluggable_database  => 'salespdb',  
        shares         => 3,  
        utilization_limit  => NULL,  
        parallel_server_limit => NULL);  
END;
```

```
BEGIN  
    DBMS_RESOURCE_MANAGER.CREATE_CDB_PLAN_DIRECTIVE(  
        plan          => 'daytime_plan',  
        pluggable_database  => 'hrpdb',  
        shares         => 1,  
        utilization_limit  => 70,  
        parallel_server_limit => 70);  
END;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Create the CDB resource plan directives for the PDBs using the `CREATE_CDB_PLAN_DIRECTIVE` procedure. Each directive specifies how resources are allocated to a specific PDB.

Here, the salespdb PDB gets three shares and default limits, hrpdb gets 1 share and 70% for both limits.

Creating CDB Resource Plan: SQL Example

```

BEGIN
  DBMS_RESOURCE_MANAGER.UPDATE_CDB_DEFAULT_DIRECTIVE(
    plan                  => 'daytime_plan',
    new_shares            => 1,
    new_utilization_limit => 50,
    new_parallel_server_limit => 50);
END;

```



```

BEGIN
  DBMS_RESOURCE_MANAGER.UPDATE_CDB_AUTOTASK_DIRECTIVE(
    plan                  => 'daytime_plan',
    new_shares            => 1,
    new_utilization_limit => 75,
    new_parallel_server_limit => 75);
END;

```



```

exec DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();

```



```

exec DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();

```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

All other PDBs in this CDB use the default PDB directive that you can optionally update using the UPDATE_CDB_DEFAULT_DIRECTIVE. In the example on the slide, you only change the limits' defaults to 50%.

Similarly, if the current autotask CDB resource plan directive does not meet your requirements, then update the directive using the UPDATE_CDB_AUTOTASK_DIRECTIVE procedure. Here, all three parameters are changed to 1 and 75% respectively.

When you are done, simply validate and submit your pending area.

Viewing CDB Resource Plan Directives

| Plan | Pluggable Database | Shares | Utilization Limit | Parallel Server Limit |
|--------------------------|----------------------------|--------|-------------------|-----------------------|
| DAYTIME_PLAN | ORA\$DEFAULT_PDB_DIRECTIVE | 1 | 50 | 50 |
| DAYTIME_PLAN | ORA\$AUTOTASK | 1 | 75 | 75 |
| DAYTIME_PLAN | SALESPDB | 3 | | |
| DAYTIME_PLAN | HRPDB | 1 | 70 | 70 |
| DEFAULT_CDB_PLAN | ORA\$DEFAULT_PDB_DIRECTIVE | 1 | 100 | 100 |
| DEFAULT_CDB_PLAN | ORA\$AUTOTASK | | 90 | 100 |
| DEFAULT_MAINTENANCE_PLAN | ORA\$DEFAULT_PDB_DIRECTIVE | 1 | 100 | 100 |
| DEFAULT_MAINTENANCE_PLAN | ORA\$AUTOTASK | | 90 | 100 |
| ORA\$INTERNAL_CDB_PLAN | ORA\$DEFAULT_PDB_DIRECTIVE | | | |
| ORA\$INTERNAL_CDB_PLAN | ORA\$AUTOTASK | | | |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This example uses the DBA_CDB_RSRC_PLAN_DIRECTIVES view to display all of the directives defined in all of the CDB resource plans in the root container.

The ORA\$INTERNAL_CDB_PLAN is the default CDB resource plan. It is a very simple plan where no resources are managed.

Note: From the root container, you can use DBA_CDB_RSRC_PLANS view to display all of the CDB resource plans defined in the root container.

Enabling a CDB Resource Plan

```
SQL> ALTER SYSTEM SET resource_manager_plan = 'daytime_plan';
```

```
BEGIN  
  DBMS_SCHEDULER.CREATE_WINDOW(  
    window_name      => 'daytime',  
    resource_plan   => 'daytime_plan',  
    start_date      => '12-Feb-01 8:00:00AM',  
    repeat_interval => 'freq=daily',  
    duration        => interval '10' hour);  
END;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You enable the Resource Manager for a CDB by setting the RESOURCE_MANAGER_PLAN initialization parameter in the root. This parameter specifies a CDB resource plan. If no plan is specified with this parameter, then the Resource Manager is not enabled.

The example in the slide uses the ALTER SYSTEM command to set the CDB resource plan to the daytime_plan.

Optionally, you can associate a CDB plan to scheduler window so that the plan is automatically set when the window opens.

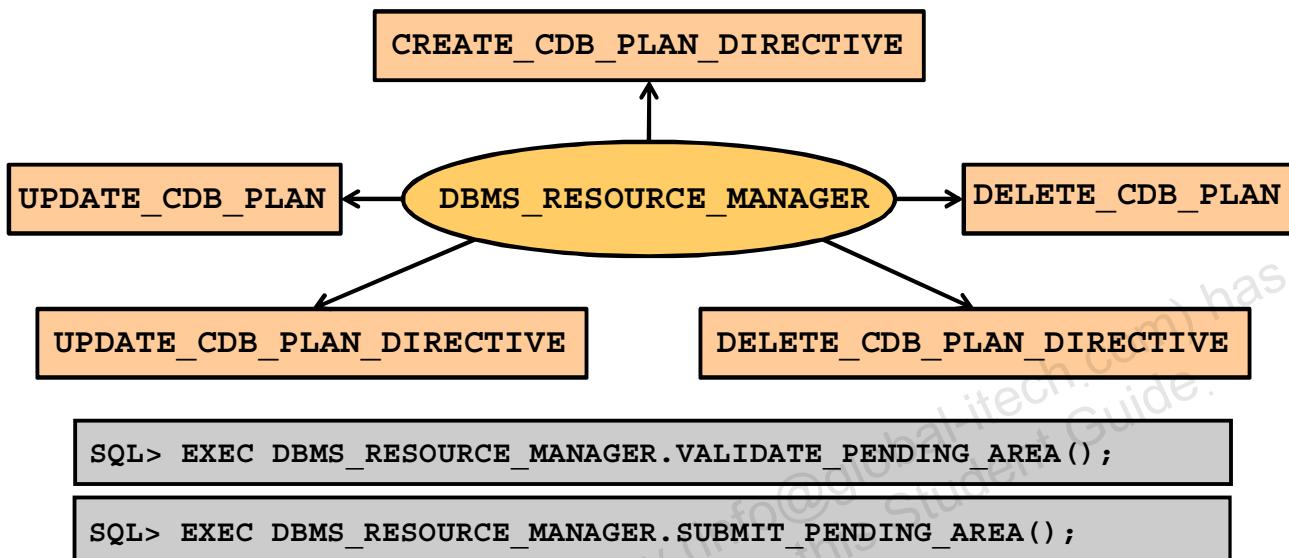
Note: You can disable CDB resource management by using the following SQL statement from the root container: ALTER SYSTEM SET RESOURCE_MANAGER_PLAN = ''

If you disable a CDB resource plan, then some directives in PDB resource plans become disabled.

Maintaining a CDB Resource Plan

```
SQL> CONNECT sys AS SYSDBA
```

```
SQL> EXEC DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
```



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

1. You can update a CDB resource plan to change its comment using the UPDATE_CDB_PLAN procedure.
2. When you create a PDB in a CDB, you can create a CDB resource plan directive for the PDB using the CREATE_CDB_PLAN_DIRECTIVE procedure. The directive specifies how resources are allocated to the new PDB.
3. You can delete the CDB resource plan directive for a PDB using the DELETE_CDB_PLAN_DIRECTIVE procedure. You might delete the directive for a PDB if you unplug or drop the PDB. However, you can retain the directive, and if the PDB is plugged into the CDB in the future, the existing directive applies to the PDB.
4. You can update the CDB resource plan directive for a PDB using the UPDATE_CDB_PLAN_DIRECTIVE procedure. The directive specifies how resources are allocated to the PDB.
5. You can delete a CDB resource plan using the DELETE_CDB_PLAN procedure. You might delete a CDB resource plan if the plan is no longer needed. You can enable a different CDB resource plan, or you can disable Resource Manager for the CDB. If you delete an active CDB resource plan, then some directives in PDB resource plans become disabled.

Managing Resources Within a PDB

- In a non-CDB database, workloads within a database are managed with resource plans.
- In a PDB, workloads are also managed with resource plans, also called PDB resource plans.
- The functionality is similar except for the following differences:

| Non-CDB Database | PDB Database |
|----------------------------|----------------------------------|
| Multi-level resource plans | Single-level resource plans only |
| Up to 32 consumer groups | Up to 8 consumer groups |
| Subplans | No subplans |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A CDB resource plan determines the amount of resources allocated to each PDB. A PDB resource plan determines how the resources allocated to a specific PDB are allocated to consumer groups within that PDB. A PDB resource plan is similar to a resource plan for a non-CDB. Specifically, a PDB resource plan allocates resource among the consumer groups within a PDB.

In a CDB, the following restrictions apply to PDB resource plans:

- PDB resource plan cannot have a multiple-level scheduling policy.
- PDB resource plan can have a maximum of eight consumer groups.
- PDB resource plan cannot have subplans.

Note: If you try to create a PDB plan in the root, you get an error.

Managing PDB Resource Plans

- Connect to the PDB to manage the PDB plan.
- Use exactly the same procedures as for managing a resource plan in a non-CDB environment.
- For CREATE_PLAN_DIRECTIVE procedure:
 - New SHARE argument
 - Replace MAX_UTILIZATION_LIMIT and PARALLEL_TARGET_PERCENTAGE arguments with UTILIZATION_LIMIT and PARALLEL_SERVER_LIMIT
- You can view CDB and PDB resource plans using V\$RSRC_PLAN.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

V\$RSRC_PLAN exists in previous releases of the Oracle Database. The resource plan with con_id=1 is the active CDB resource plan.

Putting It Together

- How do CDB and PDB resource plans work together?
 1. Resources allocated to a PDB, based on CDB resource plan
 2. Resource allocated to a consumer group based on the PDB resource plan

CDB Plan

| PDB | Shares | Utilization Limit |
|------|--------|-------------------|
| PDB1 | 1 | 50% |
| PDB2 | 1 | 50% |
| PDB3 | 2 | 100% |

PDB3 Plan

| Consumer Group | Shares | Utilization Limit |
|----------------|--------|-------------------|
| OLTP | 3 | 100% |
| REPORTS | 1 | 50% |
| OTHER | 1 | 50% |

- What does this mean for PDB3 Reports?
 - Guaranteed $50\% (2/4) \times 20\% (1/5) = 10\%$ of the resources
 - Limited to $100\% \times 50\% = 50\%$ of the resources

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Resources are allocated to each PDB based on the CDB resource plan in action. Each PDB receives a certain percentage of the system resources and Resource Manager distribute that lot to each consumer group based on that PDB's resource plan in action.

An example is shown on the slide. The example just use SHARES and UTILIZATION LIMIT for simplicity.

As you can see, the REPORTS consumer group is guaranteed to receive 10% of the total system resources available. This is because PDB3 is allocated 2 shares out of 4 in the CDB resource plan, which represents 50% of the resources, and out of this 50%, the REPORTS consumer group receives one share out of five, which represents 20% of the given 50%. Similarly, the REPORTS consumer group in PDB3 is limited to 50% of total system resources.

Considerations

- Setting a PDB resource plan is optional. If not specified, all sessions within the PDB are treated equally.
- A non-CDB is plugged into a CDB with a plan:
 - The plan runs exactly the same on the PDB if:
 - Consumer groups <= 8
 - No subplans
 - All allocations on level 1
 - The plan is converted.
 - The original plan is stored in the dictionary with the LEGACY status.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

V\$RSRC_PLAN exists in previous releases of the Oracle Database. The resource plan with con_id=1 is the active CDB resource plan.

If a non-CDB with a plan is plugged into a CDB as a new PDB, the plan acts exactly the same way in the PDB if it does not violate any of the PDBs restrictions:

- The number of consumer groups does not exceed 8.
- There are no subplans.
- All allocations are on level 1.

However, if it violates any of these restrictions, the plan is converted to something equivalent. If the plan is enabled, the converted version is used. If the converted plan does not meet your expectations, you can still work the original plan, stored in the dictionary with the LEGACY.

Runaway Queries and Resource Manager

- New parameters to trigger consumer group switching:
 - SWITCH_IO_LOGICAL
 - SWITCH_ELAPSED_TIME
- New meta consumer group called LOG_ONLY
- New column added to V\$SQL_MONITOR:
 - RM_LAST_ACTION
 - RM_LAST_ACTION_REASON
 - RM_LAST_ACTION_TIME
 - RM_CONSUMER_GROUP
- V\$RSRCMGRMETRIC and V\$RSRCMGRMETRIC_HISTORY always populated



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

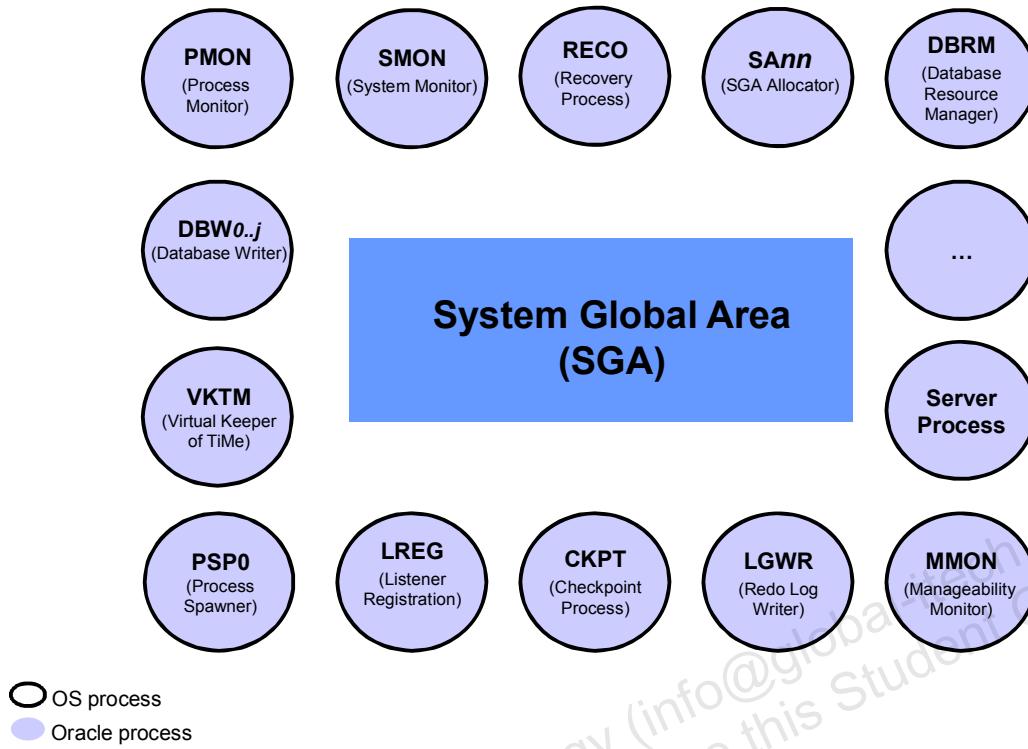
To better track runaway queries, Resource Manager is enhanced to include the following new features:

- In addition to existing switch conditions (switch_io_reqs, switch_io_megabytes, switch_time), new parameters are added to dbms_resource_manager.create_plan_directive:
 - switch_io_logical: Number of logical I/Os that will trigger the action specified by switch_group
 - switch_elapsed_time: Elapsed time that will trigger the action specified by switch_group
- There is a new meta-consumer group called LOG_ONLY. This can be used as the argument for the switch_group parameter. It means that the DBA wants to log the runaway query without changing its consumer group or taking other action.
- Resource Manager integrates the runaway query information with SQL Monitor. To retain important results for Resource Manager, it pins up to five SQLs per consumer group. SQL Monitor does not purge these SQL executions until they are unpinned. Here are the new columns added to V\$SQL_MONITOR:
 - RM_LAST_ACTION: The most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following: CANCEL_SQL, KILL_SESSION, LOG_ONLY, SWITCH TO <CG NAME>

- RM_LAST_ACTION_REASON: The reason for the most recent action that was taken on this SQL operation by Resource Manager. Its value is one of the following:
SWITCH_CPU_TIME, SWITCH_IO_REQS, SWITCH_IO_MBS,
SWITCH_ELAPSED_TIME, SWITCH_IO_LOGICAL
- RM_LAST_ACTION_TIME: The time of the most recent action that was taken on this SQL operation by Resource Manager
- RM_CONSUMER_GROUP: The current consumer group for this SQL operation

Note: v\$rsrcmgrmetric and v\$rsrcmgrmetric_history will always produce a row every minute regardless of whether there is a Resource Manager plan set.

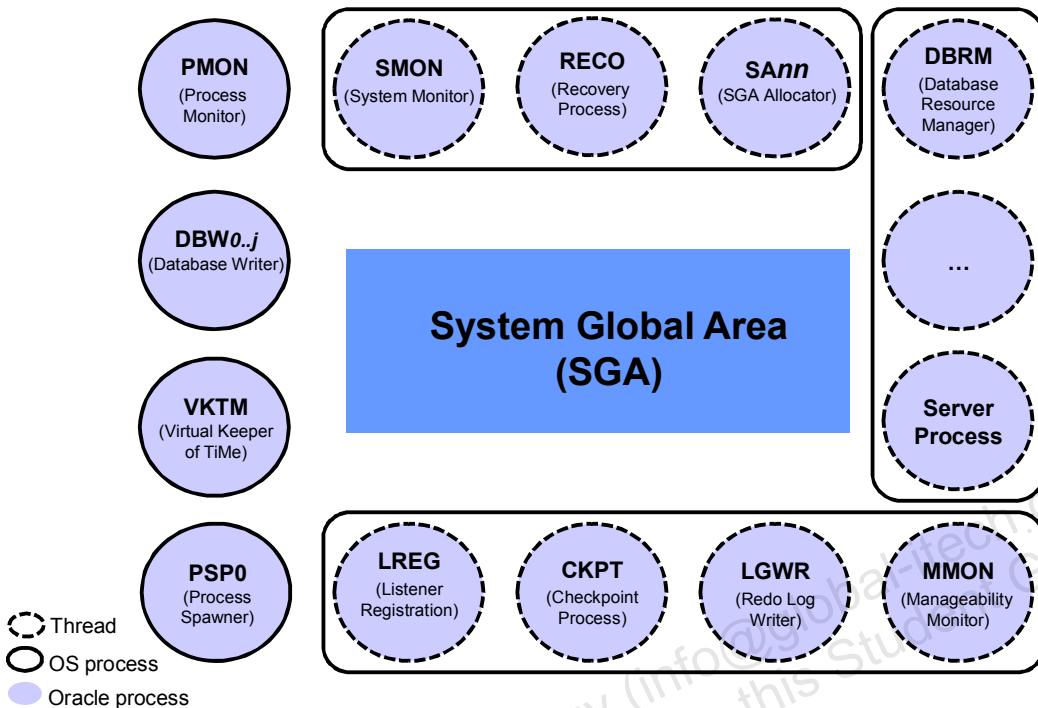
Default UNIX/Linux Architecture



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Oracle process execution architecture for an Oracle Database Instance depends on the operating system. For example, on Windows an Oracle background process is a thread of execution within an Operating System (OS) process. On Linux and UNIX, an Oracle background process runs as an OS process. This situation is represented on the slide where every Oracle background process and foreground process, generically called Oracle process, runs as a specific OS process.

Multi-Process Multi-Threaded UNIX/Linux Architecture



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Starting in Oracle Database 12c, the multithreaded Oracle model enables certain Oracle processes to execute as operating system threads in separate address spaces.

In threaded mode, some background processes on UNIX and Linux run as processes (processes containing one thread), whereas the remaining Oracle processes run as threads within OS processes. The OS process to Oracle process allocation is random based.

This is illustrated in the slide where Oracle processes PMON, DBW, VKTM, and PSP run in their own OS process, while all the others run as threads in different OS processes.

Note

- As of Oracle Database 12c release 12.1, and because of their importance, it was decided to always have PMON, DBW, VKTM, and PSP running as OS processes. This can change in future releases.
- Each OS process also runs a special thread called SCMN that is basically an internal listener thread. All thread creation is routed through this thread. This special thread is not shown on the slide.

Multi-Process Multi-Threaded Architecture: Benefits

- CPU usage reduction
- Memory usage reduction
- Better system reliability
- Better performance for parallel operations



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

In some tests, memory usage was almost divided by half when using multi-process multi-threaded feature.

Multi-Process Multi-Threaded Architecture Setup

```
SQL> CONNECT sys AS SYSDBA
Enter password:
Connected.
SQL> ALTER SYSTEM SET threaded_execution=true SCOPE=SPFILE;

System altered.

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



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Setting up an Oracle database instance for using multi-process multi-threaded architecture is done by starting up the Oracle database instance with the initialization parameter THREADED_EXECUTION set to TRUE.

Note: To be able to start up an Oracle database instance with THREADED_EXECUTION set to TRUE, you need to use a SYSDBA connection that was authenticated via a password file.

Multi-Process Multi-Threaded Architecture Considerations

- Threads still use PGA for private memory.
- Threads still use SGA memory for inter-process communication.
- The listener does not spawn threads.
- Password file for SYSDBA authentication is required.
- As a best practice, set the following parameters:

```
LOCAL_LISTENER = <Instance's TNS name entry>
```

```
DEDICATED_THROUGH_BROKER_LISTENER = on
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- An Oracle process running as a thread still uses PGA for private memory.
- An Oracle process running as a thread still uses SGA memory for inter-process communication.
- The database listener does not spawn threads, instead the connection requests, handled by a dispatcher, are handed out to the spawning thread for the given instance. Each OS process running multiple Oracle processes also run an internal listener thread called SCMN. All thread creation is routed through this thread.
- Multi-process Multi-threaded mode requires password file for SYSDBA authentication while starting the database instance. Without using SYS password, an ORA-1031: insufficient privileges error is triggered on startup.
- When connecting to the database instance using a TNS name (from `tnsnames.ora`), by default, an OS process is started to run the Oracle server process as a thread within that OS process. To make sure that the database connection spawns a thread and not an OS process, you need to set the `LOCAL_LISTENER` initialization parameter to a TNS name entry corresponding to your instance service, and set the `DEDICATED_THROUGH_BROKER_LISTENER` parameter to `ON` in your `listener.ora` file. When that parameter is set, the listener knows that it should not spawn an OS process when a connect request is received, instead it passes the request to the database so that a database thread is spawned and answers the connection.

Multi-Process Multi-Threaded Architecture Monitoring

```
SQL> select spid, stid, pname from v$process order by spid;
```

| SPID | STID | PNAME |
|-------------------|-------|-------|
| 31562 | 31562 | PMON |
| 31566 | 31566 | PSP0 |
| 31570 | 31570 | VKTM |
| 31576 | 31580 | GEN0 |
| 31576 | 31627 | SMON |
| 31576 | 31633 | LREG |
| 31576 | 31624 | LG01 |
| ... | | |
| 31590 | 31608 | DIA0 |
| 31590 | 31590 | SCMN |
| 31590 | 31642 | D000 |
| ... | | |
| 31598 | 31602 | OFSD |
| 31598 | 31598 | SCMN |
| 31612 | 31612 | DBW0 |
| 47 rows selected. | | |

```
SQL>
```

```
[oracle@host01 ~]$ ps -ef | grep orcl
oracle      598 19126  0 16:30 pts/0  00:00:00 grep orcl
oracle    31562      1  0 14:04 ?        00:00:01 ora_pmon_orcl
oracle    31566      1  0 14:04 ?        00:00:01 ora_psp0_orcl
oracle    31570      1  1 14:04 ?        00:01:52 ora_vktm_orcl
oracle    31576      1  0 14:04 ?        00:00:05 ora_u004_orcl
oracle    31576      1  0 14:04 ?        00:00:35 ora_u005_orcl
oracle    31590      1  0 14:04 ?        00:00:00 ora_u006_orcl
oracle    31598      1  0 14:04 ?        00:00:00 ora_dbw0_orcl
...
[oracle@host01 ~]$
[oracle@host01 ~]$ ps -eLo "pid tid comm args" | grep ora_
31562 31562 ora_pmon_orcl  ora_pmon_orcl
31566 31566 ora_psp0_orcl  ora_psp0_orcl
...
31570 31570 ora_vktm_orcl  ora_vktm_orcl
31576 31576 ora_scmn_orcl  ora_u004_orcl
31576 31579 oracle          ora_u004_orcl
31576 31580 ora_gen0_orcl  ora_u004_orcl
31576 31583 ora_mman_orcl  ora_u004_orcl
[oracle@host01 ~]$
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Database Smart Flash Cache Enhancements

- Multiple flash drives in the flash cache
 - Dynamic enable/disable for flash cache devices
- Change to in-memory parallel query (PQ) algorithm
 - When Auto DOP is enabled
 - Buffer cache divided into OLTP and DW sections
 - Separate cache replacement algorithm used for DW section



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 11g R2, the Smart Flash Cache was limited to one device. For systems where there was a need for a flash cache larger than could be provided by a single flash device ASM diskgroup or a volume manager was required to present the multiple devices as a single device or file.

In Oracle Database 12c, up to 16 devices can be specified to be part of the flash cache. These devices can be enabled and disabled dynamically, but resizing the flash cache dynamically is not supported.

In Oracle Database 12c, an object based algorithm has been added to the standard LRU algorithm for managing the database buffer cache with a flash cache to allow in-memory parallel query to take advantage of the flash cache. The specialized object-based policy segregates the buffer cache, including the flash cache, with a portion of the buffers being replaced by the specialized object-based algorithm, while the other portion being replaced by the traditional LRU algorithm, which also include the original smart flash cache algorithm. In the RAC environment, if a large object is smaller than the combined size of the combined buffer caches and flash caches size for the cluster, the object is partitioned and fit into memory and flash completely. With in-memory parallel query, it can eliminate disk reads for this query, or intelligently read from disks in parallel to increase the read bandwidth.

Note: This new algorithm is enabled when using automatic degree of parallelism (Auto DOP) by setting the initialization parameter `PARALLEL_DEGREE_POLICY` to `AUTO`.

Enabling and Disabling Flash Devices

- Specify flash devices at instance startup.

```
db_flash_cache_file = /dev/raw/sda, /dev/raw/sdb  
db_flash_cache_size = 32G, 64G
```

- Disable a flash device.

```
db_flash_cache_size = 0, 64G
```

- Enable a flash device.

```
db_flash_cache_size = 32G, 64G
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, you specify the flash devices and their sizes at instance startup in the SPFILE or test parameter file. Even though you cannot change the size of the flash cache dynamically, you can enable and disable individual devices as shown.

The DB_FLASH_CACHE_FILE parameter is not dynamic and can only be set at instance startup. Specify a comma separated list of up to 16 OS devices. The example shown is appropriate for a Linux system.

In this example, the flash device /dev/raw/sda is initialized at startup with 32 GB. With the ALTER SYSTEM SET command you can disable /dev/raw/sda by setting the size to 0. You can also enable /dev/raw/sda back to the same size it was previously that is 32 GB. You cannot change the size of the device using this method, or change the devices specified.

If there is only one device in the list of devices and its size is set to 0, the flash cache is disabled.

In-Memory PQ Algorithm: Benefits

- Allows parallel query to utilize smart flash cache
- Allows smart flash cache and disk to be read in parallel
- Does not thrash the buffer cache or flash cache
- Seeks an optimal balance between IO and CPU resources



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The in-memory PQ object replacement algorithm is used to support Data Warehouse scalability by making use of the flash cache on a compute node in a large cluster. The advantages of this algorithm are also apparent in smaller instances that run a mixed workload, or a workload that changes from OLTP to DSS at times. This algorithm does not replace the LRU algorithm, but exists in the same instance.

This algorithm:

- Allows parallel query to utilize smart flash cache
- Reads the smart flash cache and disk in parallel
- Will not force already cached parts of a object out of the cache for more of the same object
- Will not force buffer blocks being used by the main LRU algorithm out of the buffer cache or flash cache
- Seeks an optimal balance between I/O and CPU resources

Smart Flash Cache: New Statistics

Several new statistics to measure the use of the object replacement algorithm

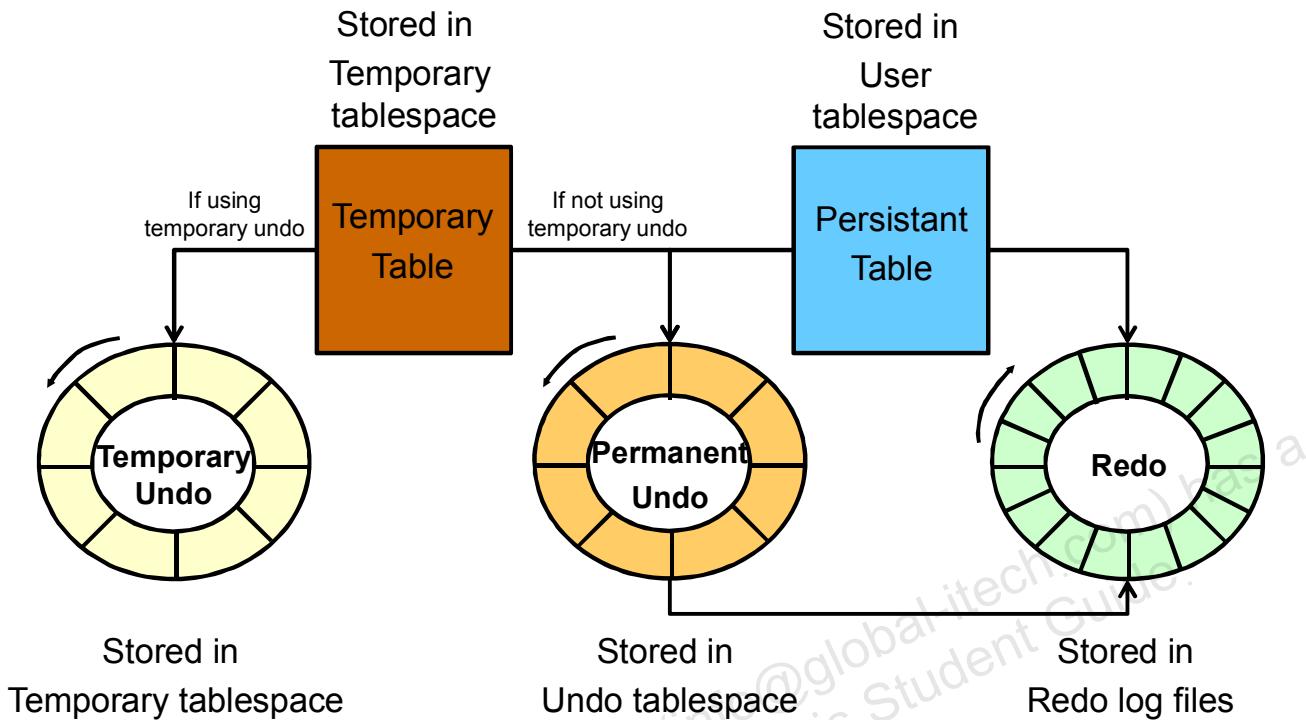
- Data warehousing scanned chunks
- Data warehousing scanned chunks - disk
- Data warehousing scanned chunks - flash
- Data warehousing scanned chunks - memory
- Data warehousing scanned objects



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- **Data warehousing scanned chunks:** Number of 1 MB chunks that have been scanned using the 12c in-memory PQ data warehousing scan
- **Data warehousing scanned chunks - disk:** Number of 1 MB chunks that have been scanned using the 12c in-memory PQ data warehousing scan, and the request was satisfied by direct read on disk
- **Data warehousing scanned chunks - flash:** Number of 1 MB chunks that have been scanned using the 12c in-memory PQ data warehousing scan, and the request was satisfied by buffers in Database Smart Flash Cache
- **Data warehousing scanned chunks - memory:** Number of 1 MB chunks that have been scanned using the 12c in-memory PQ data warehousing scan, and the request was satisfied by buffers in memory
- **Data warehousing scanned objects:** Number of objects that have been scanned using the 12c in-memory PQ data warehousing scan

Temporary Undo: Overview



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

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 due to the fact that no redo entries are directly generated for changes on temporary tables. However, the undo for operations on temporary tables (and indexes) 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. Hence 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 new mode is called temporary undo.

Note: 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

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



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Enabling temporary undo provides the following benefits:

- Temporary undo reduces the amount of undo stored in the undo tablespaces. Less undo in the undo tablespaces can result in more realistic undo retention period requirements for undo records.
- Temporary undo reduces the size of the redo log. Performance is improved because less data is written to the redo log, and components that parse redo log records, such as LogMiner, perform better because there is less redo data to parse.
- Temporary undo enables data manipulation language (DML) operations on temporary tables in a physical standby database with the Oracle Active Data Guard option. However, data definition language (DDL) operations that create temporary tables must be issued on the primary database.

Temporary Undo Setup

- Setup:

```
ALTER SESSION SET TEMP_UNDO_ENABLED = TRUE;
```

```
ALTER SYSTEM SET TEMP_UNDO_ENABLED = TRUE;
```

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



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can enable temporary undo for a specific session or for the whole system. 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 database with the `COMPATIBLE` initialization parameter set to at least `12.1.0.0.0`.

Note: However, 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.

Temporary Undo Monitoring

```

SELECT to_char(BEGIN_TIME, 'dd/mm/yy hh24:mi:ss') ,
       TXNCOUNT, MAXCONCURRENCY, UNDOBLKCNT, USCOUNT, NOSPACEERRCNT
  FROM V$TEMPUNDOSTAT;

TO_CHAR(BEGIN_TIM TXNCOUNT MAXCONCURRENCY UNDOBLKCNT USCOUNT NOSPACEERRCNT
----- ----- ----- ----- ----- -----
...
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>

```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

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 the 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*.

Summary

In this lesson, you should have learned how to:

- Use Resource Manager in a CDB environment
- Describe the multi-process multi-threaded Oracle architecture
- Describe Database Smart Flash Cache enhancements
- Use temporary undo for your temporary tables



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Practice 17 Overview: Using Resource Manager to Manage Pluggable Database Resources

These practices cover the following topics:

- Using Resource Manager to handle resources shared in a CDB and its PDBs
- Setting up and monitoring multi-process multi-threaded architecture



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

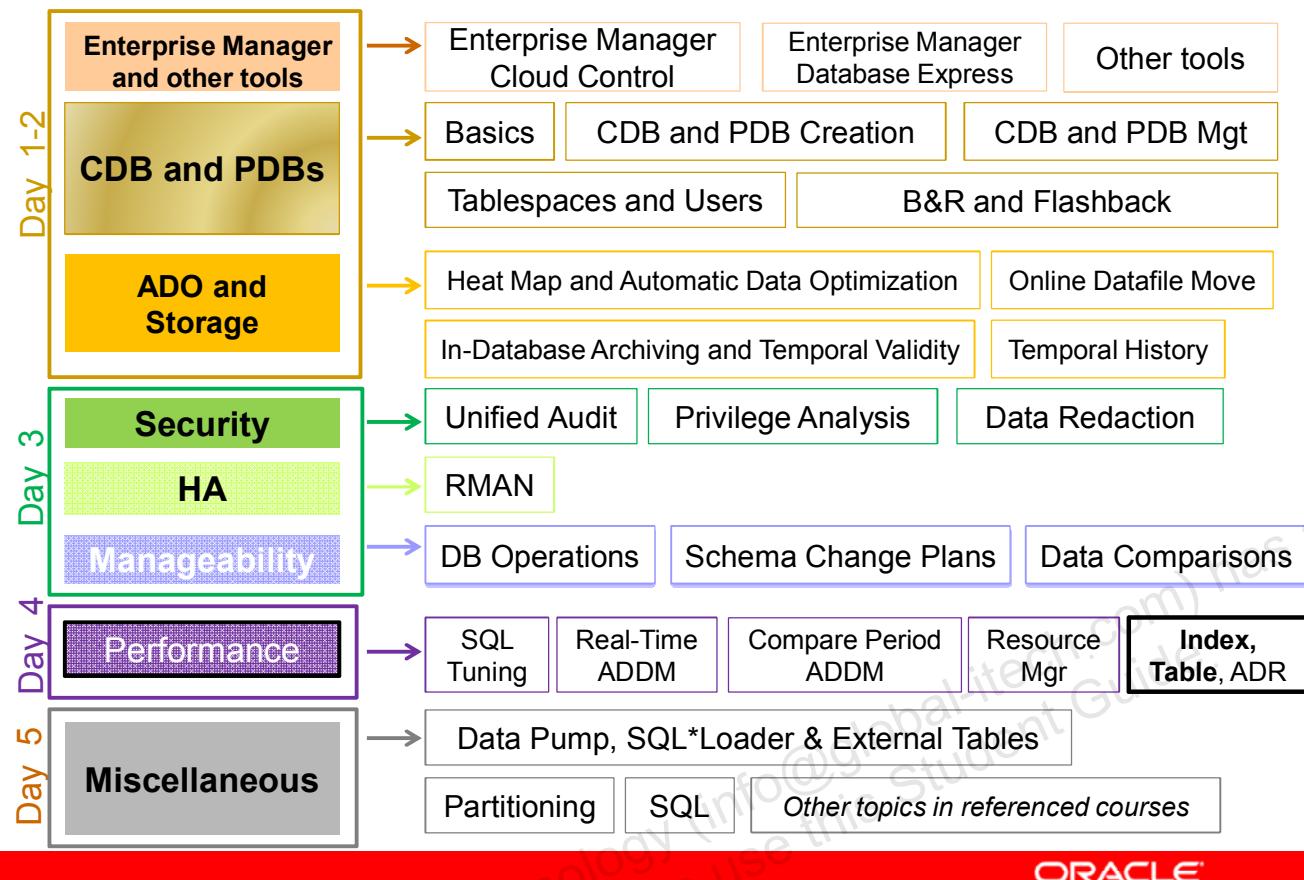
18

Tables, Indexes, and Online Operations Enhancements

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are new features for tables and indexes, like invisible columns and multiple indexes on the same set of columns, compression and Online Operations enhancements.

Objectives

After completing this lesson, you should be able to:

- Explain multiple indexes on the same set of columns
- Describe the support for invisible columns/hidden columns
- Describe online redefinition enhancements
- Describe the Advanced Row Compression
- Make more DDL statements non-blocking



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of multiple indexes on the same sets of columns and invisible columns on tables, refer to the following guide in the Oracle documentation:

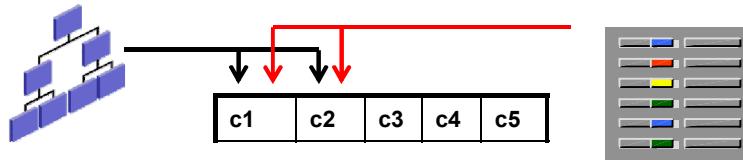
- *Oracle Database Administration Guide 12c Release 1 (12.1)*
 - *"Managing Tables" Chapter*
 - *"Managing Indexes" Chapter*

For a complete understanding of compression on tables, refer to the following guide in the Oracle documentation:

- *Oracle Database Concepts 12c Release 1 (12.1)*
 - *"Tables and Table Clusters" Chapter – Table Compression*
- *Oracle Database PL/SQL Packages and Types Reference*.

Why Multiple Indexes on the Same Set of Columns?

- Several different indexes:
 - B-tree index and bitmap index



- B-tree indexes often used in OLTP systems with many concurrent transactions
- Bitmap indexes often used in data warehousing systems that are mostly used for queries
- Different partitioning: Local and global
- Flexibility to meet your requirements:
 - Perform application migrations without dropping an existing index and recreating it with different attributes

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

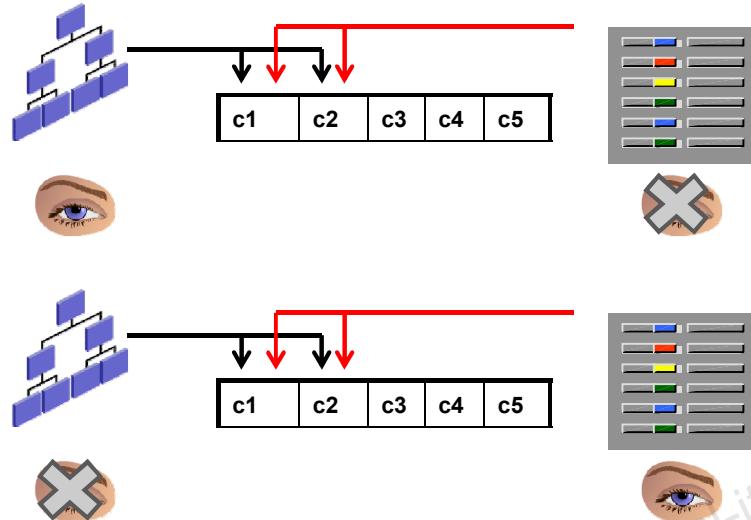
You can create multiple indexes on the same set of columns when at least one of the following index characteristics is different:

- The indexes are of different types.
However, the following exceptions apply:
 - You cannot create a B-tree index and a B-tree cluster index on the same set of columns.
 - You cannot create a B-tree index and an index-organized table on the same set of columns.
- The indexes use different partitioning:
 - Indexes that are not partitioned and indexes that are partitioned
 - Indexes that are locally partitioned and indexes that are globally partitioned
 - Indexes that differ in partitioning type (range or hash)
- The indexes have different uniqueness properties: You can create both a unique and a non-unique index on the same set of columns.

When you have multiple indexes on the same set of columns, only one of these indexes can be visible at a time.

Creating Multiple Indexes on the Same Set of Columns

Only **one** of the indexes **visible** at a time.



If `OPTIMIZER_USE_INVISIBLE_INDEXES=TRUE`, the optimizer uses the invisible index.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When you have multiple indexes on the same set of columns, only one of these indexes can be visible at a time.

If you are creating a visible index, then any existing indexes on the set of columns must be invisible. Alternatively, you can make the other indexes on the same set of columns invisible or create the index on the set of columns as invisible.

If `OPTIMIZER_USE_INVISIBLE_INDEXES` is set to true, the optimizer can use the invisible index to create a plan.

```
SQL> create table T1 as select * from hr.employees;
SQL> create index t1_i1 on T1 (employee_id) invisible;
SQL> create index t1_i2 on T1 (employee_id) reverse;
SQL> alter session set optimizer_use_invisible_indexes = true;
SQL> explain plan for select employee_id, last_name from T1 where
employee_id=100;
```

Execution Plan

Plan hash value: 2242215931

| Id Operation | Name |
|---|--------------|
| 0 SELECT STATEMENT | |
| 1 TABLE ACCESS BY INDEX ROWID BATCHED | T1 |
| * 2 INDEX RANGE SCAN | T1_I1 |

Quiz

When multiple indexes on the same set of columns exist, the optimizer chooses the best one to improve the query performance.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

Only one of the multiple indexes on the same set of columns is visible. The optimizer has therefore no choice between several indexes. The optimizer can only decide to use the visible one or not.

Invisible and Hidden Columns in SQL*Plus

- Invisible columns are user-specified hidden columns.

```
SQL> CREATE TABLE mytab (col1 VARCHAR2(10),
  2                                col2 NUMBER INVISIBLE);
```

```
SQL> DESC mytab
```

| Name | Null? | Type |
|------|-------|---------------|
| COL1 | | VARCHAR2 (10) |

The invisible column is
not displayed

- Can be made visible later
- The hidden columns do not affect existing applications that access the table.
- This hidden column can be accessed only by referring to its name explicitly in queries and other operations.

```
SQL> INSERT INTO mytab (col1, col2) values ('A',1);
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Invisible columns are introduced so that developers can enhance applications to meet constant changes in business rules and requirements. Invisible columns are user-specified hidden columns. With invisible columns, users can access the application while the developer is enhancing it to meet new business requirements. You expose the invisible column after the new business requirement is incorporated into the application. You can add a hidden column to a table, and then make it visible later on with an ALTER TABLE MODIFY statement.

Users can seamlessly add a hidden column to the table without affecting existing applications that access the table. You can use an INVISIBLE column as a partitioning key when it is specified as a part of CREATE TABLE. You can specify the invisible column as a virtual column. You cannot implicitly specify a value for an INVISIBLE column in the VALUES clause of an INSERT statement. You must specify the INVISIBLE column in the column list. You can access this hidden column only by referring to its name explicitly in queries and other operations.

Note: Invisible columns are not the same as system-generated hidden columns.

You can make invisible columns visible, but you cannot make hidden columns visible.

```
SQL> ALTER TABLE mytab MODIFY (col2 VISIBLE);
```

SET COLINVISIBLE and DESCRIBE Commands

The SET COLINVISIBLE command displays the invisible columns in a DESCRIBE command.

```
SQL> SET COLINVISIBLE ON
```

```
SQL> DESC mytab
```

| Name | Null? | Type |
|---------------------------|-------|---------------|
| COL1 | | VARCHAR2 (10) |
| COL2 (INVISIBLE) | | NUMBER |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The SET COLINVISIBLE command enables users to display the invisible column existence in a DESCRIBE command. The syntax has an ON/OFF option. ON means that metadata for the invisible column is displayed in the DESCRIBE command, and OFF means that no metadata is displayed.

When a table does not contain invisible columns, the COLINVISIBLE flag does not affect the DESCRIBE command.

The code example shows the use of the SET COLINVISIBLE command and the effect on the DESCRIBE result.

Quiz

The `SET COLINVISIBLE` command does not allow you to display the invisible columns definition in a `DESCRIBE` command.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

When set to `ON`, the command `SET COLINVISIBLE` displays the invisible columns definition.

Online Redefinition: Tables with VPD

Redefine a table with virtual private database (VPD) policies without changing the properties of any of the table's columns.

1. Apply a VPD policy on HR.EMP table using DBMS_RLS.ADD_POLICY procedure

```
SQL> EXEC DBMS_RLS.ADD_POLICY ( object_schema => 'hr', -
 2   object_name => 't1', policy_name => 't1_pol', -
 3   function_schema => 'sys', policy_function=> 'auth_sal', -
 4   statement_types => 'select, insert, update, delete')
```

2. Start the redefinition process:

- a. Verify that the table is a possible candidate for online redefinition.
- b. Create the interim table.
- c. Start the redefinition.

```
SQL> EXEC DBMS_REDEFINITION.START_REDEF_TABLE ( uname => 'hr', -
 2   orig_table => 't1', int_table => 'int_t1', -
 3   options_flag => DBMS_REDEFINITION.CONS_USE_PK, -
 4   copy_vpd_opt => DBMS_REDEFINITION.CONS_VPD_AUTO);
```

3. Complete the redefinition process.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You apply a Virtual Private Database (VPD) policy on a table to restrict the data selected or modified by users during SELECT and DML operations.

If the original table being redefined has VPD policies, you can use the `copy_vpd_opt` parameter in the `DBMS_REDEFINITION.START_REDEF_TABLE` procedure to handle these policies during online redefinition.

The possible values for the `copy_vpd_opt` parameter are the following:

- `DBMS_REDEFINITION.CONS_VPD_NONE`: Specify this value if there are no VPD policies on the original table. This value is the default. If this value is specified and VPD policies exist for the original table, then an error is raised.
- `DBMS_REDEFINITION.CONS_VPD_AUTO`: Specify this value to copy the VPD policies automatically from the original table to the new table during online redefinition. Only the table owner and the user invoking online redefinition can access the interim table during online redefinition.
- `DBMS_REDEFINITION.CONS_VPD_MANUAL`: Specify this value to copy the VPD policies manually from the original table to the new table during online redefinition.
 - There are VPD policies specified for the original table, and there are column mappings between the original table and the interim table.
 - You want to add or modify VPD policies during online redefinition of the table.

Online Redefinition: `dml_lock_timeout`

The `dml_lock_timeout` parameter in the `FINISH_REDEF_TABLE` procedure specifies how long the procedure waits for pending DML to commit.

1. Verify that the table can be redefined online.

```
SQL> EXEC DBMS_REDEFINITION.CAN_REDEF_TABLE (...);
```

2. Create the interim table.
3. Start the redefinition.

```
SQL> EXEC DBMS_REDEFINITION.START_REDEF_TABLE (...);
```

4. Copy dependent objects.

```
SQL> EXEC DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS (...);
```

5. Complete the redefinition process.

```
SQL> EXEC DBMS_REDEFINITION.FINISH_REDEF_TABLE (uname => 'hr', -  
2      orig_table      => 't1', int_table => 'int_t1'-  
3      dml_lock_timeout => 100);
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When you complete the online redefinition process, you execute the `FINISH_REDEF_TABLE` procedure. During this procedure, the original table is locked in exclusive mode for a very short time, independent of the amount of data in the original table. However, `FINISH_REDEF_TABLE` will wait for all pending DML to commit before completing the redefinition.

You can use the `dml_lock_timeout` parameter in the `FINISH_REDEF_TABLE` procedure to specify how long the procedure waits for pending DML to commit. The parameter specifies the number of seconds to wait before the procedure ends gracefully.

- When you specify a non-NULL value for this parameter, you can restart the procedure, and it continues from the point at which it timed out.
- When the parameter is set to NULL, the procedure does not time out. In this case, if you stop the procedure manually, then you must abort the online table redefinition using the `ABORT_REDEF_TABLE` procedure and start over from the beginning of step 5 (as shown in the slide).

Advanced Row Compression

– New Feature Name and Syntax –

| 11g COMPRESS | | | 12c ROW STORE COMPRESS | | |
|-------------------|------------------------|--------------|-----------------------------|-----------------------------|--------------|
| Method | CREATE and ALTER TABLE | Typical Apps | Method | CREATE and ALTER TABLE | Typical Apps |
| Basic | COMPRESS [BASIC] | DSS | Basic | ROW STORE COMPRESS | DSS |
| OLTP | COMPRESS FOR OLTP | OLTP, DSS | Advanced | ROW STORE COMPRESS ADVANCED | OLTP, DSS |

| | |
|--|--|
| SQL> CREATE TABLE tab1 COMPRESS [BASIC]; | SQL> CREATE TABLE tab1 ROW STORE COMPRESS [BASIC]; |
| SQL> CREATE TABLE tab1 COMPRESS FOR OLTP; | SQL> CREATE TABLE tab1 ROW STORE COMPRESS ADVANCED; |

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 11g extended the compression technology to support regular data manipulation operations such as INSERT, UPDATE, and DELETE. Consequently, compression could be used for all kinds of workload, be it online transaction processing (OLTP) or data warehousing.

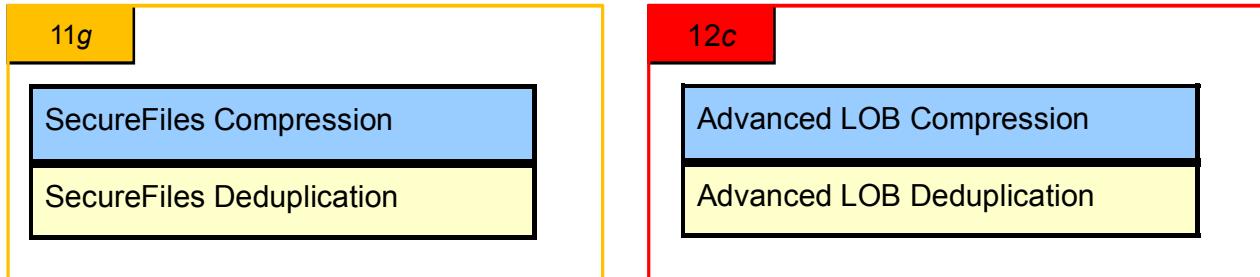
Oracle Database 12c brings incremental enhancements for OLTP Compression. The OLTP Compression feature was released in 11g Release 1 as a part of Advanced Compression Option (ACO). The OLTP Compression now named Advanced Row Compression increases savings and provides higher compression ratios. One problem with OLTP and BASIC Compression is that when a row is updated, all the columns of the row are uncompressed up to the highest referenced column for the update. Now only those columns that are updated are uncompressed, and not any columns unnecessarily. When OLTP compression did not support more than 255 columns, row store compression supports more than 255 columns. The shrink space operation is now supported for row store compressed table.

Row Store or Column Store are attributes of a table, like the organization clause. Compression and locking options are options of the store

ROW STORE and COMPRESS are the new attributes to compress a table (ROW STORE versus COLUMN STORE or HCC).

ADVANCED and BASIC are types of compression, used uniformly for tables, indexes, and LOBs to convey that it is part of ACO.

LOB Compression: New Name



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

SecureFile LOB

With Oracle Database 12c, the name of two features in the Advanced Compression Option related to SecureFiles has changed:

- SecureFiles Compression is now called Advanced LOB Compression.
- SecureFiles Deduplication is now called Advanced LOB Deduplication.

Using the Compression Advisor

- A compression advisor helps to determine optimal compression ratios.
- The DBMS_COMPRESSION package includes the GET_COMPRESSION_RATIO procedure.

```
BEGIN
    DBMS_COMPRESSION.GET_COMPRESSION_RATIO (
        'USERS', 'HR', 'EMP', NULL, DBMS_COMPRESSION.COMP_ADVANCED,
        blkcnt_cmp, blkcnt_ncmp, row_cmp, row_ncmp, cmp_ratio,
        comptype_str,1000,1);
    DBMS_OUTPUT.PUT_LINE('Block count compressed = ' || blkcnt_cmp);
    DBMS_OUTPUT.PUT_LINE('Block count uncompressed = ' || blkcnt_ncmp);
    DBMS_OUTPUT.PUT_LINE('Row count per block compressed = ' || row_cmp);
    DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed=' || row_ncmp);
    DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
    DBMS_OUTPUT.PUT_LINE('Comp ratio= '|| blkcnt_ncmp/blkcnt_cmp||' to 1');
END;
```

- New ADVANCED compression type in COMPRESS_FOR column in DBA_TABLES



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

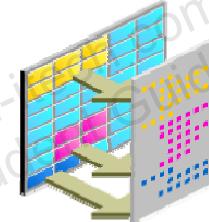
The compression advisor has been enhanced to display the new advanced compression type.

```
SQL> set serveroutput on
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 (
    'USERS', 'HR', 'EMP', NULL, DBMS_COMPRESSION.COMP_ADVANCED,
    blkcnt_cmp, blkcnt_ncmp, row_cmp, row_ncmp, cmp_ratio,
comptype_str,1000,1);
DBMS_OUTPUT.PUT_LINE('Block count compressed = ' || blkcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Block count uncomp = ' || blkcnt_ncmp);
DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype_str);
DBMS_OUTPUT.PUT_LINE('Ratio = '||blkcnt_ncmp/blkcnt_cmp||' to 1');
END;
/
Block count compressed = 5
Block count uncompressed = 7
Compression type = "Compress Advanced"
Compression ratio = 1.4 to 1
PL/SQL procedure successfully completed.
```

Enhanced Online DDL Capabilities

You can use the new **ONLINE** keyword to allow the execution of DML statements during the following DDL operations:

- DROP INDEX
- DROP CONSTRAINT
- ALTER INDEX UNUSABLE
- SET COLUMN UNUSED



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Beginning with Oracle Database 12c Release 1 (12.1), you can use the new **ONLINE** keyword to allow the execution of DML statements during the following DDL operations:

- DROP INDEX
- DROP CONSTRAINT
- ALTER INDEX UNUSABLE
- SET COLUMN UNUSED

This enhancement enables simpler application development, especially for application migrations. There are no application disruptions for schema maintenance operations.

DROP INDEX / CONSTRAINT



- **ONLINE** indicates that DML operations on the table or partition are allowed while dropping the index.
- The `DROP INDEX` online is supported for partitioned and non-partitioned indexes.

```
SQL> DROP INDEX schema.index ONLINE FORCE;
```

- The `DROP CONSTRAINT` online enables you to drop an integrity constraint from a database with some restrictions:
 - Cannot drop a constraint with `CASCADE`
 - Cannot drop a referencing constraint

```
SQL> ALTER TABLE hr.employees
  2  DROP CONSTRAINT emp_email_uk ONLINE;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `DROP INDEX` statement is used to remove an index or domain index from a database.

Drop index online is supported for partitioned and non-partitioned indexes.

ONLINE: Specify `ONLINE` to indicate that DML operations on the table or partition are allowed while dropping the index.

FORCE: `FORCE` applies only to domain indexes. This clause drops the domain index even if the index type routine invocation returns an error or if the index is marked `IN PROGRESS`. Without `FORCE`, you cannot drop a domain index if its index type routine invocation returns an error or if the index is marked `IN PROGRESS`.

Restrictions on dropping indexes:

- You cannot drop a domain index if the index or any of its index partitions is marked `IN_PROGRESS`.
- You cannot specify the `ONLINE` clause when dropping a domain index, a cluster index, or an index on a queue table.

Restrictions on dropping constraints:

- You cannot drop a constraint with `CASCADE`.
- You cannot drop a constraint that holds references for other dependencies.

Index UNUSABLE

- Specify the **UNUSABLE** clause to mark the index or index partitions or index subpartitions **UNUSABLE**.
- Specify the **ONLINE** clause to indicate that DML operations on the table or partition are allowed while marking the index **UNUSABLE**.

```
SQL> ALTER INDEX hr.i_emp_ix UNUSABLE ONLINE;
```

```
SQL> SELECT status FROM user_indexes  
2 WHERE table_name='EMP';
```

| INDEX_NAME | STATUS |
|------------|----------|
| I_EMP_IX | UNUSABLE |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Specify the **UNUSABLE** keyword to mark the index or index partitions or index subpartitions **UNUSABLE**. The space allocated for an index or index partition or subpartition is freed immediately when you mark the object **UNUSABLE**. An unusable index must be rebuilt, or it must be dropped and re-created, before it can be used. While one partition is marked **UNUSABLE**, the other partitions of the index are still valid. As of Oracle 12c, the **ALTER INDEX UNUSABLE** clause is made online by specifying an **ONLINE** keyword to indicate that DML operations on the table or partition are allowed while marking the index **UNUSABLE**. If you specify this clause, the database does not drop the index segments.

Note: You cannot specify **UNUSABLE** for an index on a temporary table.

SET UNUSED Column

- The SET UNUSED clause can be the first step to free space in the database by dropping columns no longer needed.
- The **ONLINE** keyword indicates that DML operations on the table are allowed while marking the columns UNUSED.

```
SQL> CREATE TABLE emp (ename VARCHAR2(20), id NUMBER);
SQL> INSERT INTO emp VALUES('Tim', 4);
SQL> SELECT * FROM emp;
SQL> ALTER TABLE emp SET UNUSED (ename) ONLINE;
SQL> DESC emp;
Name          Null?    Type
-----        -----
ID           NUMBER
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The drop_column_clause can be the first step to free space in the database by dropping columns that you no longer need or by marking them to be dropped at a future time when the demand on system resources is less.

You specify the SET UNUSED keyword to mark one or more columns as unused. When you specify this clause for a column in an external table, the clause is transparently converted to an ALTER TABLE ... DROP COLUMN statement. Because any operation on an external table is a metadata-only operation, there is no difference in the performance of the two commands.

Unused columns are treated as if they were dropped, even though their column data remains in the table rows. After marking a column UNUSED, you have no access to it. A SELECT * query does not retrieve data from unused columns. In addition, the names and types of columns marked UNUSED are not displayed during a DESCRIBE, and you can add a new column to the table with the same name as an unused column.

You specify the ONLINE keyword to indicate that DML operations on the table are allowed while marking the column or columns UNUSED.

You cannot specify the ONLINE clause when marking a column with a DEFERRABLE constraint as unused. A subsequent DROP UNUSED COLUMNS physically removes all unused columns from a table, similar to a DROP COLUMN.

Summary

In this lesson, you should have learned how to:

- Explain multiple indexes on the same set of columns
- Describe the support for invisible columns/hidden columns
- Describe online redefinition enhancements
- Describe the Advanced Row Compression
- Make more DDL statements non-blocking

Practice 18 Overview: Tables and Indexes Enhancements

This practice covers the topic of using invisible/visible columns.

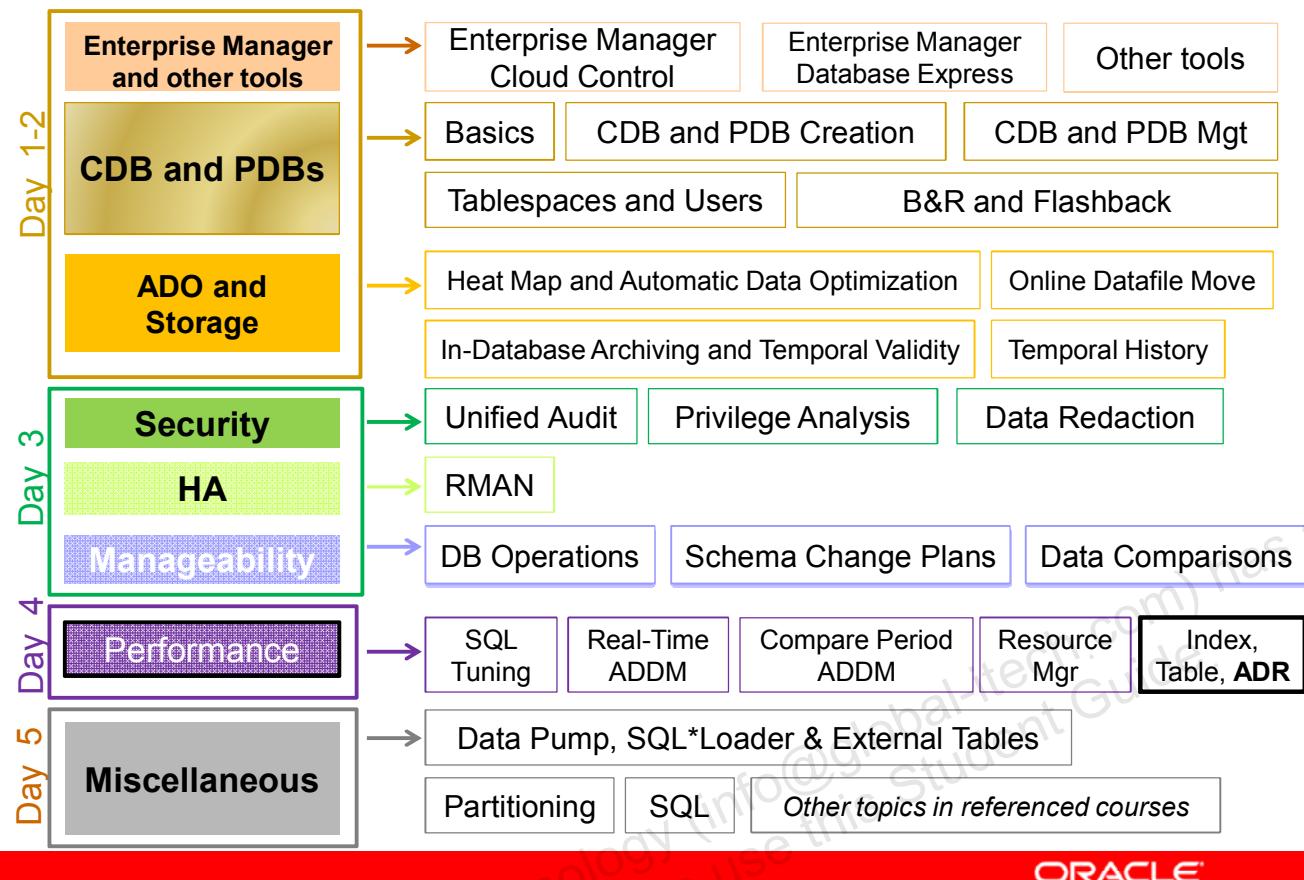
ADR and Network Enhancements

19

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several **Automatic Diagnostic Repository** (ADR) and **Network** enhancements.

Objectives

After completing this lesson, you should be able to:

- Describe the new ADR DDL and debug log files
- List and view log files using ADRCI utility commands
- Describe benefits of network data compression
- Show change in DEFAULT_SDU_SIZE



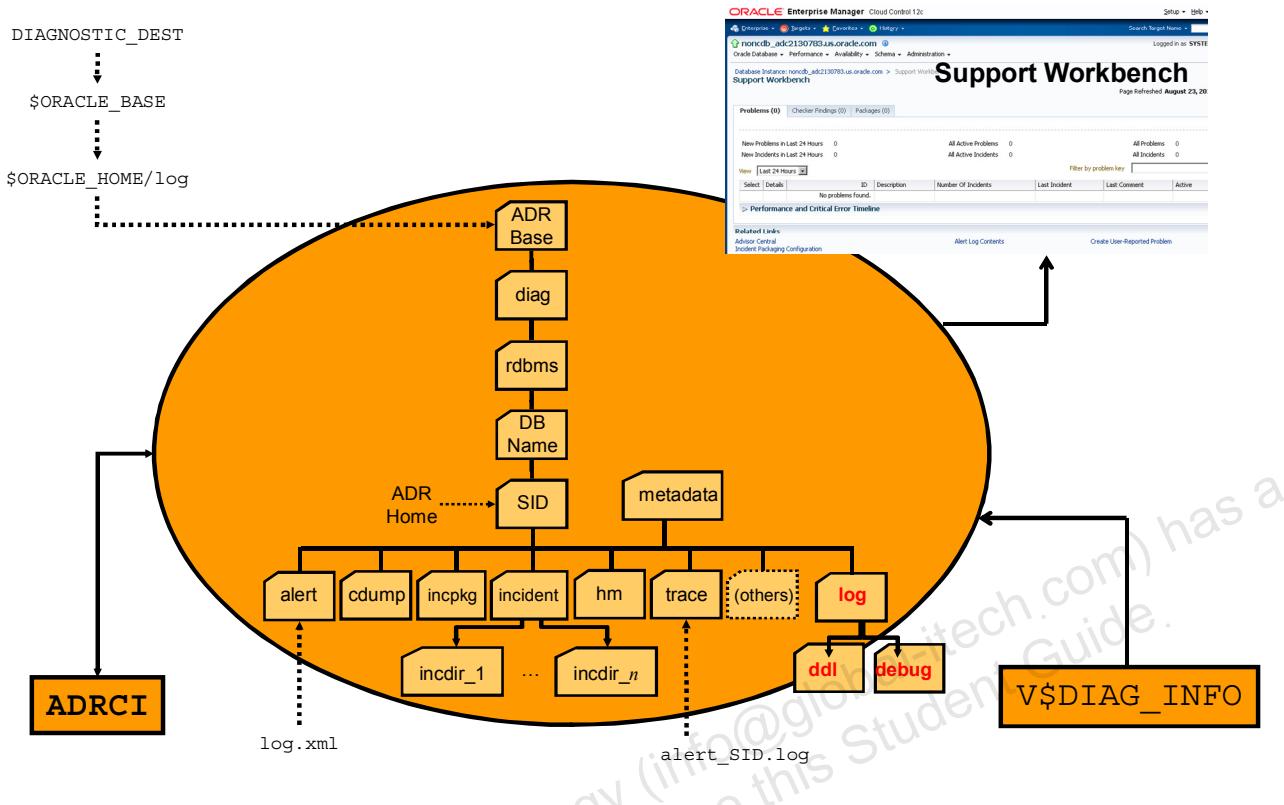
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of the Automatic Diagnostic Repository enhancements and usage, refer to the following guide in the Oracle documentation:

- *Oracle Database Administration Guide 12c Release 1 (12.1) – "Managing Diagnostic Data" Chapter*
- *Oracle Enterprise Manager Licensing Information 12c Release 1 (12.1) – "Enterprise Database Management" Chapter – "Legacy: Change Management Pack for Oracle Database" Section*
- *Oracle Database Utilities 12c Release 1 (12.1) – "ADRCI: ADR Command Interpreter" Chapter*
- *Oracle Database Net Services Administrator's Guide 12c Release 1 (12.1) – "Optimizing Performance" Chapter*
- *Oracle Database Net Services Reference 12c Release 1 (12.1) – "Parameters for the sqlnet.ora File" Chapter*

Automatic Diagnostic Repository



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

Automatic Diagnostic Repository (ADR) is a file-based repository for database diagnostic data such as traces, incident dumps and packages, the alert log, Health Monitor reports, core dumps, and more. It has a unified directory structure across multiple instances and multiple products stored outside of any database. It is, therefore, available for problem diagnosis when the database is down. Beginning with Oracle Database 11g Release 1, the database server, Automatic Storage Management (ASM), and other Oracle products or components store all diagnostic data in ADR. Each instance of each product stores diagnostic data underneath its own ADR home directory. For example, in a Real Application Clusters environment with shared storage and ASM, each database instance and each ASM instance has a home directory within ADR. ADR's unified directory structure uses consistent diagnostic data formats across products and instances. A unified set of tools enable customers and Oracle Support to correlate and analyze diagnostic data across multiple instances.

Two alert log files are generated. You can view the alert log in text format (with the XML tags stripped) with Enterprise Manager and with the ADR Command Interpreter (ADRCI) utility.

The graphic in the slide shows you the directory structure of an ADR home.

A new `log` directory contains two subdirectories, `ddl` and `debug`.

ADR File Types

- Log: High-level information of activity
- Trace:
 - Background trace files
 - SQL trace files
- Dump:
 - Specific type of trace file
 - Detailed point-in-time information of an incident
- Core:
 - Memory dump
 - All-binary, port-specific format



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Log files are shared files that contains high-level information. The RDBMS alert log is a very good example of this kind of log.

Trace, dump, and core files contain diagnostic data that are used to investigate problems, all of them involving writing messages to files.

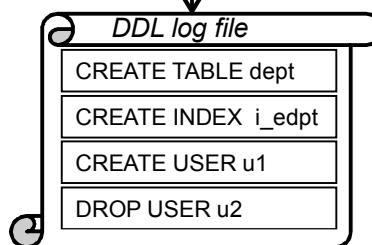
- **Trace** files contain fairly detailed information covering a long period of time, such as state transitions, or structures being worked on. Each server and background process can write to an associated trace file. Trace files are updated periodically over the life of the process and can contain information on the process environment, status, activities, and errors. In addition, when a process detects a critical error, it writes information about the error to its trace file. The SQL trace facility also creates trace files, which provide performance information on individual SQL statements.
- **Dump** files contain very detailed point-in-time information about some state or some structure. A dump is a specific type of trace file, a one-time output of diagnostic data in response to an event (such as an incident), whereas a trace tends to be continuous output of diagnostic data. When an incident occurs, the database writes one or more dumps to the incident directory created for the incident. Incident dumps also contain the incident number in the file name.
- **Core** files contain a memory dump, in an all-binary, port-specific format. Core file names include the string “core” and the operating system process ID. Core files are useful to Oracle Support engineers only. Core files are not found on all platforms.

ADR Files: DDL and DEBUG Log Files

New specific ADR log files:

- DDL log

```
SQL> ALTER SYSTEM SET enable_ddl_logging=TRUE;
```



- No DDL log file
- No DDL recording in alert log

```
SQL> ALTER SYSTEM SET enable_ddl_logging=FALSE;
```

- Debug log

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

DDL log

Oracle Database 11g added some support for DDL logging of RDBMS DDL statements into the alert log. Setting the instance parameter `ENABLE_DDL_LOGGING` to `TRUE` activates DDL logging. Oracle Database 12c actually turns DDL logging off by default setting `ENABLE_DDL_LOGGING` to `FALSE`. If turned on, the RDBMS DDL logging is written to a new ADR file type that has the same format and basic behavior as the alert log, but it only contains DDL statements and dates. The `init.ora` parameter `ENABLE_DDL_LOGGING` is licensed as part of the Change Management Pack for Oracle Database when set to `TRUE`.

Debug Log

An Oracle Database component can detect conditions, states or events that are unusual, but which do not inhibit correct operation of the detecting component. The component can issue a warning about these conditions, states, or events. These warnings are not serious enough to warrant an incident or a write to the alert log. They do warrant a record in a log file because they might be needed to diagnose a future problem. Developers may find it useful to create a record of such events, but currently lack appropriate mechanism. The debug log is a file that records these warnings. The debug log has the same format and basic behavior as the alert log, but it only contains information about possible problems that might need to be corrected, reducing the amount of information in the alert log and trace files.

The debug log is included in IPS incident packages. The debug log's contents are intended for Oracle Support. Database administrators should not use the debug log directly.

ADR Files: Location

| Diagnostic Data | ADR Location |
|---------------------------|------------------------------|
| Foreground process traces | \$ADR_HOME/trace |
| Background process traces | \$ADR_HOME/trace |
| Alert log data | \$ADR_HOME/alert&trace |
| Core dumps | \$ADR_HOME/cdump |
| Incident dumps | \$ADR_HOME/incident/incdir_n |
| DDL logs | \$ADR_HOME/log/ddl |
| Debug logs | \$ADR_HOME/log/debug |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

New ADRCI Command

Show DDL log file content:

```
adrci> SHOW LOG;
```

- The `$ADR_HOME/log/ddl/log.xml` file content is displayed with an editor:
 - vi for Unix
 - Notepad for Windows

```
2012-08-23 15:01:00.200000 +00:00
create table t(a varchar(40), b number, c varchar(240), d varchar(240))
2012-08-23 15:03:49.121000 +00:00
create table scott.tabjfv(c number) tablespace users
2012-08-23 15:11:58.017000 +00:00
drop user test cascade
~
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To view the content of the DDL log file which is in XML format, you can use the ADRCI `SHOW LOG` command.

This command automatically opens the `ddl` log file using the `vi` editor on UNIX or Notepad on Windows. Leave the `vi` editor using `:q` command.

Network Performance: Compression

- Reduces data volume
- Reduces bandwidth use
- Is transparent to application
- Compression schemes
- When to use compression?



Network performance is generally limited by two factors: the bandwidth and the data volume. In simple language the bandwidth is the size of the pipe. The larger the pipe, more data bits can be pushed through it. The data volume is the number of bits that needs to transmitted. So to improve the network performance either the bandwidth must be increased or the data volume decreased. In many cases, increasing the bandwidth is not cost effective, or is not even possible. The only option is reducing the data volume.

Compression of the data in the network layer reduces the number of bits transmitted. This reduces the bandwidth used, allowing more data to be transmitted. By performing the compression in the network layer, the compression is transparent to the application.

There are several compression schemes possible. In the advanced compression option (ACO), GZIP and LZO compression schemes can be used to compress data in network layer as they have the ability to handle data unit (SDU size) separately.

Compression exchanges computation for data volume. So compression will help in situations where the network bandwidth is the bottleneck. If the operation is already CPU bound, network compression will only make the problem worse.

Setting Up Compression

- SQLNET.COMPRESSION
- SQLNET.COMPRESSION_LEVELS
- SQLNET.COMPRESSION_THRESHOLD



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Three new parameters in the sqlnet.ora file allow the DBA to set the parameters for compression operations. Network compression is available as part of Advanced Compression Option.

SQLNET.COMPRESSION: Can be set to ON or OFF to enable or disable compression. The default is OFF.

SQLNET.COMPRESSION_LEVELS: The compression levels are used at time of negotiation to verify which levels are used at both ends, and to select one level. For Database Resident Connection Pooling (DRCP), only the compression level LOW is supported. The compression level may be set to HIGH to use high CPU usage and high compression ratio, or LOW to use low CPU usage and low compression ratio. The default is LOW.

SQLNET.COMPRESSION_THRESHOLD: This parameter specifies minimum data size required to perform compression. Compression will not be done if size of data to be sent is less than this value. The default is 1024 (bytes).

SDU Size

Modify SDU size when:

- The data coming from the server is fragmented
- You are on a wide area network (WAN) that has long delays
- The packet size is consistently the same
- Large amounts of data are returned

Do not modify SDU size when:

- The application can be tuned to reduce network use
- You have a high speed network where the effect of the data transmission is negligible
- Your requests return small amounts of data from the server



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The session data unit (SDU) size is the size of the data packet sent over the network. Setting the SDU size to a larger value can improve network performance. This value can range from 512 bytes to 2 MB bytes. The default SDU for the client and a dedicated server is 8192 bytes. The default SDU for a shared server is 65535 bytes. The actual SDU size used is negotiated between the client and the server at connect time and is the smaller of the two values.

After trace files statistics are collected with `trcasst -s` utility, consider changing the SDU size when the predominant message size is not equal to 8192. A larger message forces the network layer to split the message into multiple packets, called fragments. The SDU size should be 70 bytes larger than the predominant message size. If the predominant message size plus 70 bytes exceeds the maximum SDU, then the SDU should be set such that the message size is divided into the smallest number of equal parts where each part is 70 bytes less than the SDU size. To change the SDU size, change the `DEFAULT_SDU_SIZE` parameter in the `sqlnet.ora` file.

For example, if the majority of the messages sent and received by the application are smaller than 8 KB, taking into account the 70 bytes for overhead, then setting the SDU to 8 KB will probably produce good results. If sufficient memory is available, using the larger values for the SDU reduces the number of system calls and overhead for Oracle Net Services.

Before you change the SDU size, tune the application to use less network bandwidth. Do not change the SDU size: if you are using a high speed network where the latency of network transmission is very small, or if only small amounts of data are transmitted across the network.

Setting SDU Size

DEFAULT_SDU_SIZE in sqlnet.ora file:

- Default is 8 KB.
- Maximum is now **2 MB** from 64 KB.
- Example: DEFAULT_SDU_SIZE=4096



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The DEFAULT_SDU_SIZE parameter in the sqlnet.ora file specifies the overall SDU size for a server or a client. When the configured values of client and database server do not match for a session, the lower of the two values is used, thus the setting should be the same on both. You can override the default setting of the SDU by specifying the SDU parameter in the connect descriptor for a client.

The optimal SDU size depends on the network characteristics, data generation rate, and message size. One case where a larger SDU could help is when the sender data generation rate is much lower than the network transmission rate.

The parameter DEFAULT_SDU_SIZE is used to set this value, in Oracle Database 11g the default value is 8 KB.

Note: on a network that is near capacity larger packets can produce more collisions, reducing performance.

Quiz

DDL commands can be logged in the alert log and in a new DDL log log.xml file.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

DDL commands are logged or not. If `ENABLE_DDL_LOGGING` is set to `TRUE`, DDL commands are logged in ADR repository in the `log/ddl` directory in `log.xml` file. If `ENABLE_DDL_LOGGING` is set to `FALSE`, DDL commands are not logged at all.

Summary

In this lesson, you should have learned how to:

- Describe the new ADR DDL and debug log files
- List and view log files using ADRCI utility commands
- Describe the benefits of network data compression
- Show change in DEFAULT_SDU_SIZE

Practice 19 Overview: ADR Enhancements

This practice covers the topic of showing ADR DDL log file and content.

Module

Miscellaneous



ORACLE

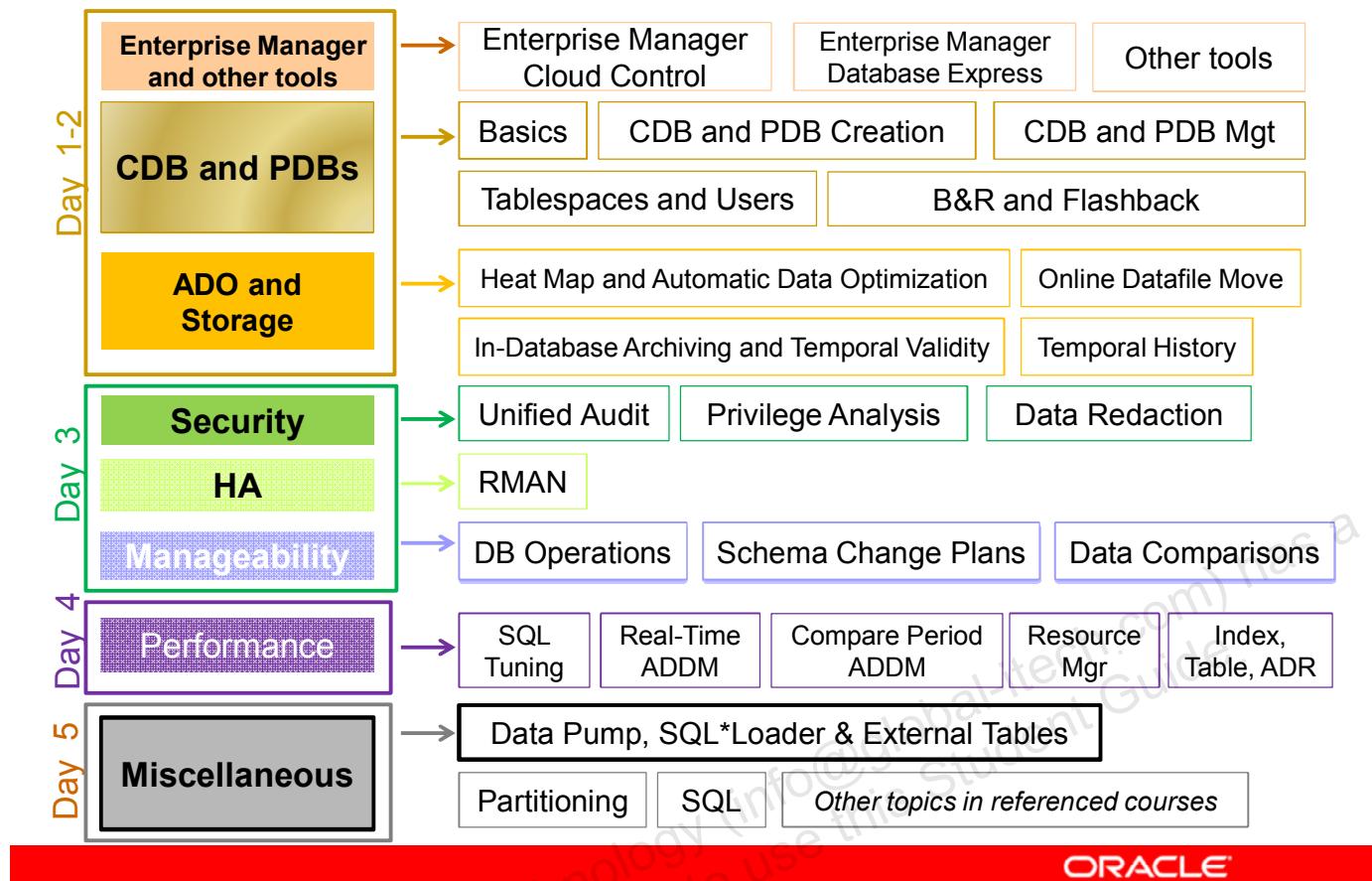
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

20 **Oracle Data Pump, SQL*Loader, and External Tables**

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several **Oracle Data Pump**, **SQL*Loader**, and **External Tables** enhancements. You will also examine **SQL*Loader Express Mode** feature.

Objectives

After completing this lesson, you should be able to:

- Describe Oracle Data Pump Full Transportable
- Describe each Oracle Database 12c new feature for Data Pump, SQL*Loader, and External Tables
- Describe applicable use cases and configuration details for the Oracle Data Pump and SQL*Loader new features
- Describe SQL*Loader Express Mode



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of Oracle Data Pump new features and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Utilities 12c Release 1 (12.1)*

Full Transportable Export/Import: Overview

- ➡ Upgrading to a new release of Oracle Database
- ➡ Transporting a database to a new computer system

- Transport a non-CDB (Oracle Database 11.2.0.3 and up) into another non-CDB.
- Transport a non-CDB (11.2.0.3 and +) into a CDB as a PDB.
- Transport a PDB into another PDB.
- Transport a PDB into a non-CDB.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Full Transportable Export/Import

The full transportable export/import feature is useful in several scenarios:

- **Upgrading to a new release of Oracle Database:** You can use full transportable export/import to upgrade a database from 11.2.0.3 or higher to Oracle Database 12c. To do so, install Oracle Database 12c and create an empty database. Next, use full transportable export/import to transport the 11.2.0.3 database into the Oracle Database 12c database.
- **Transporting a database to a new computer system:** You can use full transportable export/import to move a database from one computer system to another. You might want to move a database to a new computer system to upgrade the hardware or to move the database to a different platform.
- **Transporting a non-CDB into a non-CDB or CDB:** Transported into a CDB, the transported database becomes a PDB associated with the CDB. Full transportable export/import can move an 11.2.0.3 or higher database into an Oracle Database 12c database efficiently.

Full Transportable Export/Import: Usage

A full transportable export exports all objects and data necessary to create a complete copy of the database.

- TRANSPORTABLE=ALWAYS parameter
- FULL parameter

```
$ expdp user_name FULL=y DUMPFILE=expdat.dmp  
        DIRECTORY=data_pump_dir TRANSPORTABLE=always  
        VERSION=12.0 LOGFILE=export.log
```

A full transportable import imports a dump file only if it has been created using the transportable option during export.

- TRANSPORT_DATAFILES
- If NETWORK_LINK used, requires:
 - TRANSPORTABLE=ALWAYS parameter



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Full Transportable Export

A full transportable export exports all objects and data necessary to create a complete copy of the database. A mix of data movement methods is used:

- Objects residing in transportable tablespaces have only their metadata unloaded into the dump file set. The data itself is moved when you copy the data files to the target database. The data files that must be copied are listed at the end of the log file for the export operation.
- Objects residing in non-transportable tablespaces (for example, SYSTEM and SYSAUX) have both their metadata and data unloaded into the dump file set, using direct path unload and external tables.

The example shows a full transportable export using the VERSION parameter. The VERSION parameter is required if the source database is an 11.2.0.3 or a higher Oracle Database 11g release.

Performing a full transportable export has the following restrictions:

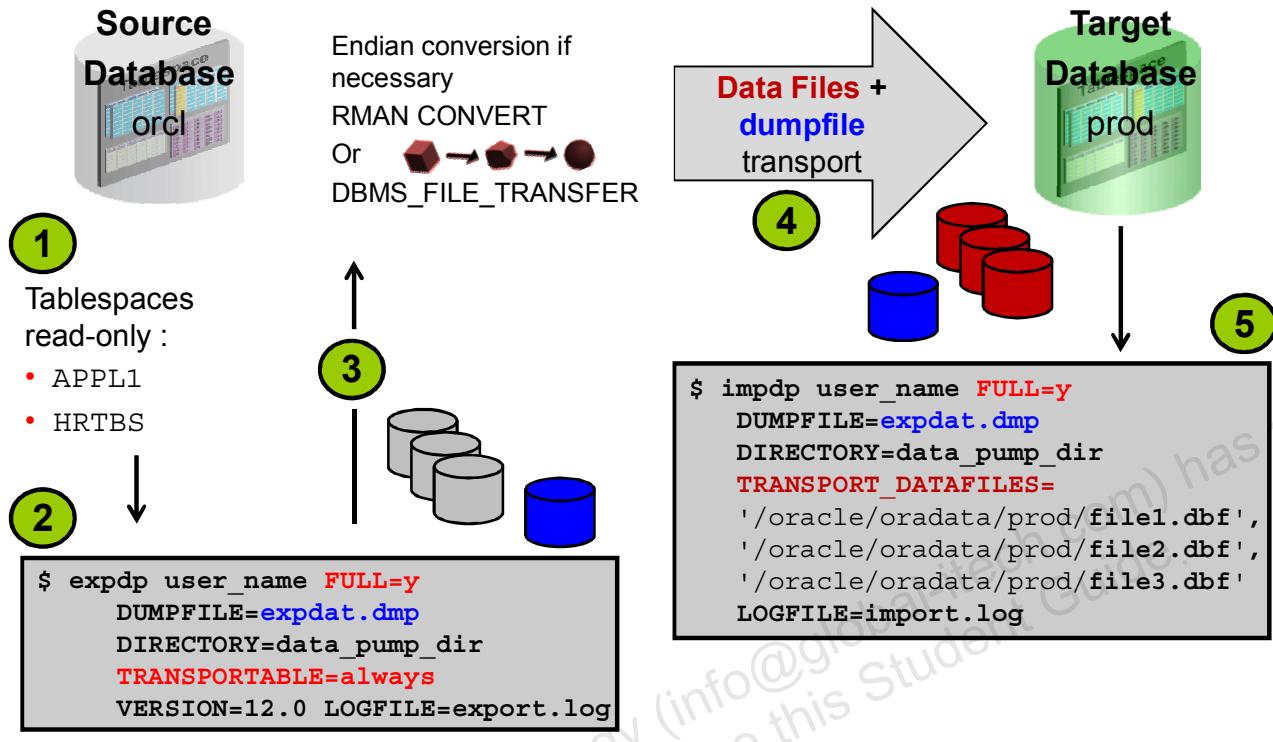
- If the database being exported contains either encrypted tablespaces or tables with encrypted columns (either Transparent Data Encryption (TDE) columns or SecureFile LOB columns), then the ENCRYPTION_PASSWORD parameter must also be supplied.
- The source and target databases must be on platforms with the same endianness if there are encrypted tablespaces in the source database.
- If the source platform and the target platform are of different endianness, you must convert the data being transported so that it is in the format of the target platform. Use either the DBMS_FILE_TRANSFER package or the RMAN CONVERT command.
- A full transportable export is not restartable.
- All objects with storage that are selected for export must have all of their storage segments either entirely within administrative, non-transportable tablespaces (SYSTEM / SYSAUX) or entirely within user-defined, transportable tablespaces. Storage for a single object cannot straddle the two kinds of tablespaces.
- When transporting a database over the network using full transportable export, tables with LONG or LONG RAW columns that reside in administrative tablespaces (such as SYSTEM or SYSAUX) are not supported.
- When transporting a database over the network using full transportable export, auditing cannot be enabled for tables stored in an administrative tablespace (such as SYSTEM and SYSAUX) if the audit trail information itself is stored in a user-defined tablespace.
- If both the source and target databases are running Oracle Database 12c Release 1 (12.1), then to perform a full transportable export, either the Oracle Data Pump VERSION parameter must be set to at least 12.0. or the COMPATIBLE database initialization parameter must be set to at least 12.0 or later.
- Full transportable exports are supported from an 11.2.0.3 source database.

Full Transportable Import

Performing a full transportable import has the following requirements:

- If you are using a network link, then the database specified on the NETWORK_LINK parameter must be Oracle Database 11g release 2 (11.2.0.3) or later, and the Oracle Data Pump VERSION parameter must be set to at least 12. (In a non-network import, VERSION=12 is implicitly determined from the dump file.)
- If the source platform and the target platform are of different endianness, then you must convert the data being transported so that it is in the format of the target platform. You can use the DBMS_FILE_TRANSFER package or the RMAN CONVERT command to convert the data.
- A full transportable import of encrypted tablespaces is not supported in network mode or dump file mode if the source and target platforms do not have the same endianess.
- When transporting a database over the network using full transportable import, tables with LONG or LONG RAW columns that reside in administrative tablespaces (such as SYSTEM or SYSAUX) are not supported.
- When transporting a database over the network using full transportable import, auditing cannot be enabled for tables stored in an administrative tablespace (such as SYSTEM and SYSAUX) if the audit trail information itself is stored in a user-defined tablespace.

Full Transportable Export/Import: Example



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To perform a Full Transportable operation, perform the following steps:

1. Before the export, make all of the user-defined tablespaces in the database read-only.
2. Invoke the Oracle Data Pump export utility as a user with DATAPUMP_EXP_FULL_DATABASE role and specify the full transportable export options: FULL=Y, TRANSPORTABLE=ALWAYS.
The LOGFILE parameter is important because it will contain the list of data files that need to be transported for the import operation.
To perform the operation on an 11.2.0.3 database, use the VERSION parameter. Full transportable import is supported only for Oracle Database 12c databases.
3. Before the import, transport the dump file.
4. Transport the data files that you may have converted. If you are transporting the database to a platform different from the source platform, then determine if cross-platform database transport is supported for both the source and target platforms. If both platforms have the same endianness, no conversion is necessary. Otherwise you must do a conversion of each tablespace in the database either at the source or target database using either DBMS_FILE_TRANSFER or RMAN CONVERT command.
5. Invoke the Oracle Data Pump import utility as a user with DATAPUMP_IMP_FULL_DATABASE role and specify the full transportable import options: FULL=Y, TRANSPORT_DATAFILES.
6. Make the source tablespaces read-write. You can perform this operation before step 5.

Transporting a Database Over the Network: Example

Transport a database over the network: perform an import using the NETWORK_LINK parameter.

1. Create a database link in the target to the source database.
2. Make the user-defined tablespaces in the source database read-only.
3. Transport the datafiles for all of the user-defined tablespaces from the source to the target location.
4. Perform conversion of the datafiles if necessary.
5. Import in the target database.

```
$ impdp username full=Y network_link=sourcedb  
      transportable=always  
      transport_datafiles='/oracle/oradata/prod/sales01.dbf',  
                        '/oracle/oradata/prod/cust01.dbf'  
      encryption_password=pass1 version=12 logfile=import.log
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To transport a database over the network, use import with the NETWORK_LINK parameter. The import is performed using a database link, and there is no dump file involved.

- If the source database is an 11.2.0.3 database or a higher Oracle Database 11g release database, then the VERSION parameter is required and must be set to 12. If the source database is an Oracle Database 12c database, then the VERSION parameter is not required.
- If the source database contains any encrypted tablespaces or tablespaces containing tables with encrypted columns, then you must specify the ENCRYPTION_PASSWORD parameter.
- The Oracle Data Pump network import copies the metadata for objects contained within the user-defined tablespaces and both the metadata and data for user-defined objects contained within the administrative tablespaces, such as SYSTEM and SYSAUX.
- When the import is complete, the user-defined tablespaces are in read-write mode.
- Make the user-defined tablespaces read-write again at the source database.

Disabling Logging for Oracle Data Pump Import: Overview

- The goal is to deliver faster data loads:
 - Redo log information is not written to disk.
 - Greater I/O bandwidth is dedicated to loading.
- This option is particularly useful for:
 - Performing large data loads
 - Populating new databases
- A full database backup is recommended after the data load finishes.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Data Pump provides an option to disable logging during an import operation. The result is quicker imports, because redo log information is not written to disk or archived, allowing greater I/O bandwidth to be dedicated to data loading. Disabling logging is particularly useful for large data loads or jobs that initially populate a new database.

If you disable logging, Oracle recommends that you perform a full Recovery Manager (RMAN) backup after the Oracle Data Pump import operation is completed. Also, in the unlikely event of media failure in the middle of the import, you must restart the import.

Disabling Logging for Oracle Data Pump Import: Usage

- Command-line syntax:

```
$ impdp ... TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y|N[:TABLE|INDEX]
```

- Notes:

- Logging is not completely eliminated. A small amount of log activity is still generated by the import.
- Logging is not disabled if the database is in FORCE LOGGING mode.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can disable logging using a new metadata transform parameter, DISABLE_ARCHIVE_LOGGING. An outline of the syntax is shown in the slide and some examples follow on the next page.

If set to `y`, then the logging attributes for the specified object types, TABLE and/or INDEX are disabled before the data is imported. If set to `n` (the default), then archive logging is not disabled during import. After the data has been loaded, the logging attributes for the objects are restored to their original settings. If no object type is specified, then the DISABLE_ARCHIVE_LOGGING behavior is applied to both TABLE and INDEX object types.

Note that the operations against the master table that is used by Oracle Data Pump to coordinate its activities are logged, even if logging is disabled for the Oracle Data Pump import session.

If the database is in FORCE LOGGING mode, logging is not disabled during the Oracle Data Pump load even if the option to do so is specified.

Disabling Logging for Oracle Data Pump Import: Command-Line Examples

```
$ impdp hr/hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp SCHEMAS=hr  
TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y
```

1

```
$ impdp hr/hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp SCHEMAS=hr  
TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y:INDEX
```

2

```
$ impdp hr/hr DIRECTORY=dpump_dir1 DUMPFILE=hr.dmp SCHEMAS=hr  
TRANSFORM=DISABLE_ARCHIVE_LOGGING:Y  
TRANSFORM=DISABLE_ARCHIVE_LOGGING:N:TABLE
```

3



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The slide shows three `impdp` command-line examples where logging is disabled.

- Example 1 is a simple case, where logging is disabled for both TABLE and INDEX object types.
- Example 2 disables logging is disabled for INDEX only.
- Example 3 shows a different way to achieve the same outcome as example 2.

Exporting Views as Tables: Overview

- The following are exported:
 - A table definition and the view data (instead of just the view definition SQL)
 - Dependent objects (such as constraints and grants)
- This option is particularly useful for:
 - Exporting a data subset
 - Exporting a denormalized data set spanning multiple tables



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Data Pump provides the option to export a view as a table. Instead of just exporting the view definition SQL, Oracle Data Pump writes into the dump file a corresponding table definition and all the data visible in the view.

At import time, Oracle Data Pump can create a table with the same geometry (columns and data types) as the original view and populate it with the exported data.

Objects depending on the view are also exported as if they were defined on the table. For example, grants that are associated with the view are recorded as grants on the corresponding table in the export dump file. Dependent objects that do not apply to tables are not exported.

The ability to export views as tables can be used to export data subsets for development and testing, data replication, and offline archival purposes.

You can also use this feature to export a denormalized data set spanning multiple related tables. This kind of operation is commonly encountered when data is extracted from transactional systems and loaded into decision support systems and data warehouses.

Exporting Views as Tables: Usage

- Command-line syntax:

```
$ expdp ... VIEWS_AS_TABLES=[schema_name.]view_name ...
```

- Only relational views with scalar columns are supported.
- Exported data is unencrypted by default.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can export views as tables by using the command-line parameter `VIEWS_AS_TABLES`.

You can also set it programmatically in PL/SQL by using the

`DBMS_DATAPUMP.METADATA_FILTER` procedure. In both cases, you must provide a view name, which can be schema-qualified.

Exporting views as tables is limited only to relational views with scalar columns. More complex views, including views that reference object types or functions, are not supported.

The `VIEWS_AS_TABLES` parameter unloads view data in unencrypted format. If you unload sensitive data, Oracle strongly recommends that you enable encryption on the export operation and that you ensure that the imported table is encrypted.

Exporting Views as Tables: Command-Line Examples

```
$ expdp hr/hr DIRECTORY=dpump_dir1 DUMPFILE=hr1.dmp  
VIEWS_AS_TABLES=employees_v TABLES=departments
```

1

```
$ expdp hr/hr DIRECTORY=dpump_dir1 DUMPFILE=hr2.dmp  
VIEWS_AS_TABLES=managers_v,oe.orders_v
```

2

```
$ impdp hr1/hr1 DIRECTORY=dpump_dir1 DUMPFILE=hr2.dmp  
VIEWS_AS_TABLES=managers_v REMAP_TABLE=managers_v:managers
```

3

```
$ impdp hr1/hr1 VIEWS_AS_TABLES=employees_v NETWORK_LINK=hr_link  
REMAP_TABLE=employees_v:employees TABLE_EXISTS_ACTION=append
```

4

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The slide shows five examples using the VIEWS_AS_TABLES command-line parameter.

- Example 1 is a simple case, where the VIEWS_AS_TABLES parameter specifies a single view (hr.employees_v). You can use this parameter by itself or with the TABLES parameter. If you use either parameter, Oracle Data Pump performs a table-mode export.
- Example 2 exports two views as tables (hr.managers_v and oe.orders_v).
- Example 3 uses the export dump file that was created in example 2. It demonstrates using the VIEWS_AS_TABLES parameter in conjunction with impdp to selectively import managers_v as the managers table.
- Example 4 shows the VIEWS_AS_TABLES parameter being used in conjunction with a network import. In this mode, the syntax for the VIEWS_AS_TABLES parameter is the same as that supported by Oracle Data Pump export (expdp). In this example, the view employees_v is sourced from the database specified by the hr_link database link. The data from employees_v@hr_link is imported into the employees table in the hr1 schema on the target database. Because TABLE_EXISTS_ACTION=append is specified, the data from the source view rows are appended to the hr1.employees table if it already exists.

Specifying the Encryption Password

- New option prompts users to specify encryption password:

```
$ expdp ... ENCRYPTION_PWD_PROMPT=Y ...  
...  
Password: <Specify password here. Input not echoed>  
...
```

- With this feature, encryption password security is enhanced because it is supplied silently at run time.
 - The password is not visible by using commands like `ps`.
 - The password is not stored in scripts.
- Concurrent use of the `ENCRYPTION_PASSWORD` parameter and `ENCRYPTION_PWD_PROMPT=Y` is prohibited and generates an error.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Earlier versions of the Oracle Data Pump utilities (`expdp` and `impdp`) provided support for encrypting data written to the export dump file. Although you can use an Oracle Wallet for transparent encryption, password-based encryption is also available.

To support password-based encryption and enable a password to be specified, a command-line parameter, `ENCRYPTION_PASSWORD`, was added to the Oracle Data Pump utilities. Before Oracle Database 12c, using the `ENCRYPTION_PASSWORD` parameter was the only way to set a password for password-based encryption in conjunction with Oracle Data Pump export. The issue with this mechanism is that the encryption password can be easily compromised. Non-privileged system users can typically use utilities such as `ps` on Linux and UNIX to view command-line parameters, including the encryption password. Also, storing the encryption password inside scripts is generally considered to be poor practice from a security perspective.

With Oracle Database 12c, the Oracle Data Pump utilities have the ability to prompt the user for an encryption password without the value being echoed as it is entered. The new command-line parameter, `ENCRYPTION_PWD_PROMPT`, can be set to `Y` to instruct the utilities to prompt the user for the encryption password. The password is read from the standard input stream, and terminal echo is suppressed while the password is being entered.

Compressing Tables During Import

- Compression method can now be specified at import time.
 - Independent of compression settings present at export time.
- Command-line syntax and example:

```
$ impdp ... TRANSFORM=TABLE_COMPRESSION_CLAUSE:NONE | "<comp-clause>" ...
```

```
$ impdp hr/hr DIRECTORY=dpump_dir1 DUMPFILE=export.dmp
  TRANSFORM=TABLE_COMPRESSION_CLAUSE:"ROW STORE COMPRESS"
```

- Syntax notes:
 - NONE => compression clause omitted from CREATE TABLE
 - <comp-clause> => valid table compression clause



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Using earlier versions of Oracle Data Pump, tables are imported with the same compression settings that were present during export. Oracle Database 12c enables Oracle Data Pump to specify the compression method at import time regardless of the settings in the source database.

The option to compress (or uncompress) tables during import is provided by a new metadata transform parameter, TABLE_COMPRESSION_CLAUSE, which can be set on the `impdp` command line or programmatically by using the `DBMS_DATAPUMP.METADATA_TRANSFORM` PL/SQL procedure.

If the parameter is set to `NONE`, the table compression clause is omitted and imported tables get the default compression setting associated with the tablespace. This option is particularly useful when you want to apply a uniform compression method to a series of tables that may have had different compression settings in the source database.

Otherwise, the parameter value must be a valid table compression clause, such as `NOCOMPRESS`, `ROW STORE COMPRESS`, and `ROW STORE COMPRESS ADVANCED`.

Table compression clauses containing more than one keyword must be enclosed in double quotation marks when they are specified on the `impdp` command line. Because quotation marks in the command line can have special meaning in some operating systems, consider using a parameter file to set the transform parameter.

Creating SecureFile LOBs During Import

- Option to specify LOB storage method at import time
- Command-line syntax and example:

```
$ impdp ... TRANSFORM=LOB_STORAGE:SECUREFILE|BASICFILE|DEFAULT|NO_CHANGE  
...
```

```
$ impdp hr/hr DIRECTORY=dpump_dir1 DUMPFILE=export.dmp  
LOB_STORAGE:SECUREFILE
```

- Syntax notes:
 - DEFAULT => no lob storage clause for CREATE TABLE
 - NO_CHANGE => settings specified in dump file



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c enables Oracle Data Pump to specify the LOB storage method at import time regardless of the settings in the source database. This capability provides a straightforward mechanism for users to migrate from BasicFile LOBs to SecureFile LOBs.

The option to specify the LOB storage method during import is provided by a new metadata transform parameter, `LOB_STORAGE`, which can be set on the `impdp` command line or programmatically by using the `DBMS_DATAPUMP.METADATA_TRANSFORM` PL/SQL procedure.

By setting the parameter to `SECUREFILE` or `BASICFILE`, you specify the corresponding LOB storage method for all LOBs that are associated with the import session. If you set the parameter to `DEFAULT`, the LOB storage clause is omitted and imported tables get the default setting that is associated with the session or instance. If you set the parameter to `NO_CHANGE`, the LOB storage clauses that are specified in the export dump file are used.

Quiz

Setting TRANSFORM=TABLE_COMPRESSION_CLAUSE : NONE on the impdp command causes all the tables created by the impdp session to be uncompressed.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

Imported tables get the default compression setting associated with the tablespace.

SQL*Loader Support for Direct-Path Loading of Identity Columns

- An identity column is a numeric column that is assigned an integer value from a sequence generator for each `INSERT` performed on the table.
- SQL*Loader supports direct-path loading of identity columns:

| Column specification | Action performed by SQL*Loader |
|---|---|
| <code>col1 NUMBER GENERATED ALWAYS AS IDENTITY</code> | <code>col1</code> cannot be loaded explicitly. The column value is always provided by the sequence generator. |
| <code>col2 NUMBER GENERATED BY DEFAULT AS IDENTITY</code> | <code>col2</code> can be loaded explicitly. If it is not, the column value is provided by the sequence generator. Explicitly loaded <code>NULL</code> values cause the corresponding row to be rejected because identity columns implicitly carry the <code>NOT NULL</code> constraint. |
| <code>col3 NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY</code> | <code>col3</code> can be loaded explicitly. If it is not, the column value is provided by the sequence generator. Explicitly loaded <code>NULL</code> values are replaced by sequence values. |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

SQL*Loader new features start with support for direct-path loading of identity columns.

An identity column is a numeric column that is assigned an increasing or decreasing integer value from a sequence generator for each `INSERT` statement that is performed on the associated table. You specify identity columns by using the `GENERATED ... AS IDENTITY` clause in association with a `CREATE TABLE` or `ALTER TABLE` statement. Identity columns are new in Oracle Database 12c.

In conjunction with this new feature, SQL*Loader supports direct-path loading of identity columns. The table on the slide describes the actions that are performed by SQL*Loader for different types of identity columns. These actions occur automatically and are consistent with the normal behavior of identity columns.

SQL*Loader and External Table Enhancements

Wildcards allowed in names of data files

- INFILE ('emp*.dat') or ('emp?.dat')

Support for CSV files with embedded newlines

- FIELDS CSV WITH EMBEDDED ...

Table-level NULLIF and Date formats can be specified once for all character and date fields

- FIELDS DATE FORMAT "DD-Month-YYYY"
- FIELDS TERMINATED BY "," NULLIF = "NA"

With FIELD NAMES clause, the first record in the data file contains the names and order of the fields

- FIELD NAMES FIRST FILE

Specific to external tables: ALL FIELDS OVERRIDE

- Default attributes for columns so you only have to specify columns that do not match the default



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

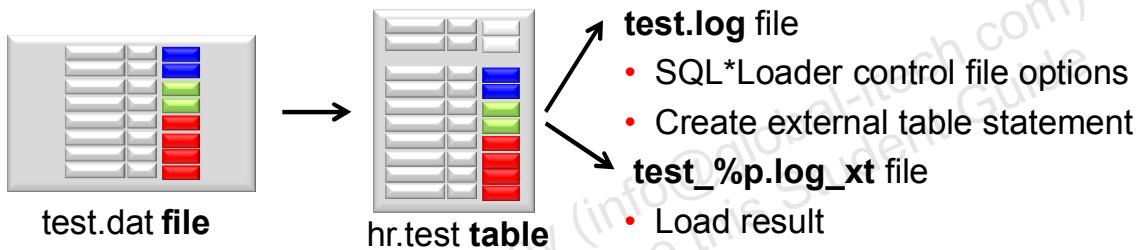
The following statements can be applied to SQL*Loader and External Tables. The syntax may differ when using SQL*Loader or External Tables.

- Wildcards are allowed in data file names: "*" is for multiple characters and "?" is for single characters.
- CSV files are supported in Oracle Database 12c under conditions. Read *Oracle Database Utilities 12c Release 1 (12.1)* to get the conditions under which CSV files data can be loaded.
- Table-level NULLIF and date format can be specified once for all fields and can be overridden for an individual field. It applies to all character and date fields for NULLIF and to all date fields for date format.
- The first record containing the names and order of the fields can be in the first file or every file.
- FIRST FILE indicates that the first data file contains a list of field names for the data in the first record. ALL FILES indicates that all data files contain a list of field names for the data in the first record. The first record is skipped in each data file when the data is processed. The fields can be in a different order in each data file.
- The last statement applies only to external tables. The ALL FIELDS OVERRIDE clause tells the access driver that all fields are present and that they are in the same order as the columns in the external table. You only need to specify fields that have a special definition.

SQL*Loader Express Mode

- It is simple to use.
- Specify a table name to initiate an Express Mode load:
 - The table columns must be scalar datatypes (character, number, or datetime).
 - The data file contains only delimited character data.
 - SQL*Loader uses table column definitions to determine input data types.
- There is no need to create a control file.

```
$ sqlldr hr TABLE=test
```



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

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

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 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.

- 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.

An example of the input data file named `test.dat` containing two records to load and excerpts of the two log files generated after the load completed. The table `HR.TEST` was created with three columns, `C1 NUMBER`, `C2 VARCHAR2(10)`, and `C3 VARCHAR2(10)`.

```
$ more test.dat
3 C WWW
4 D UUU
$ sqldr hr TABLE=test
$ more test.log
...
Express Mode Load, Table: TEST
Data File:      test.dat
    Bad File:   test_%p.bad ...
Table TEST, loaded from every logical record.
Insert option in effect for this table: APPEND
Column Name      Position  Len  Term Encl Datatype
-----
C1              FIRST      * , CHARACTER
C2              NEXT       * , CHARACTER
C3              NEXT       * , CHARACTER
Generated control file for possible reuse:
OPTIONS (EXTERNAL_TABLE=EXECUTE, TRIM=LRTRIM)
LOAD DATA INFILE 'test' APPEND INTO TABLE TEST
FIELDS TERMINATED BY "," ( C1, C2, C3)
End of generated control file for possible reuse.
...
creating external table "SYS_SQLLDR_X_EXT_TEST"
CREATE TABLE "SYS_SQLLDR_X_EXT_TEST"
("C1" NUMBER,"C2" CHAR(1),"C3" VARCHAR2(20)) ORGANIZATION external(
  TYPE oracle_loader DEFAULT DIRECTORY SYS_SQLLDR_XT_TMPDIR_00000
  ACCESS PARAMETERS
  ( RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    BADFILE 'SYS_SQLLDR_XT_TMPDIR_00000':'test_%p.bad'
    LOGFILE 'test_%p.log_xt' READSIZE 1048576
    FIELDS TERMINATED BY "," LRTRIM REJECT ROWS WITH ALL NULL FIELDS
    ("C1" CHAR(255), "C2" CHAR(255), "C3" CHAR(255)))
  location ('test.dat') ) REJECT LIMIT UNLIMITED
executing INSERT statement to load database table TEST
INSERT /*+ append parallel(auto) */ INTO TEST (C1, C2, C3)
SELECT "C1", "C2", "C3" FROM "SYS_SQLLDR_X_EXT_TEST" ...
Table TEST: 2 Rows successfully loaded.
```

Summary

In this lesson, you should have learned how to:

- Describe Oracle Data Pump Full Transportable
- Describe each Oracle Database 12c new feature for Oracle Data Pump, SQL*Loader, and External Tables
- Describe applicable use cases and configuration details for the Oracle Data Pump and SQL*Loader new features
- Describe SQL*Loader Express Mode

Practice 20 Overview: Oracle Data Pump Enhancements

These practices cover the following topics:

- Transporting a database by using the Full Transportable feature
- Loading data by using SQL*Loader Express Mode (*optional*)

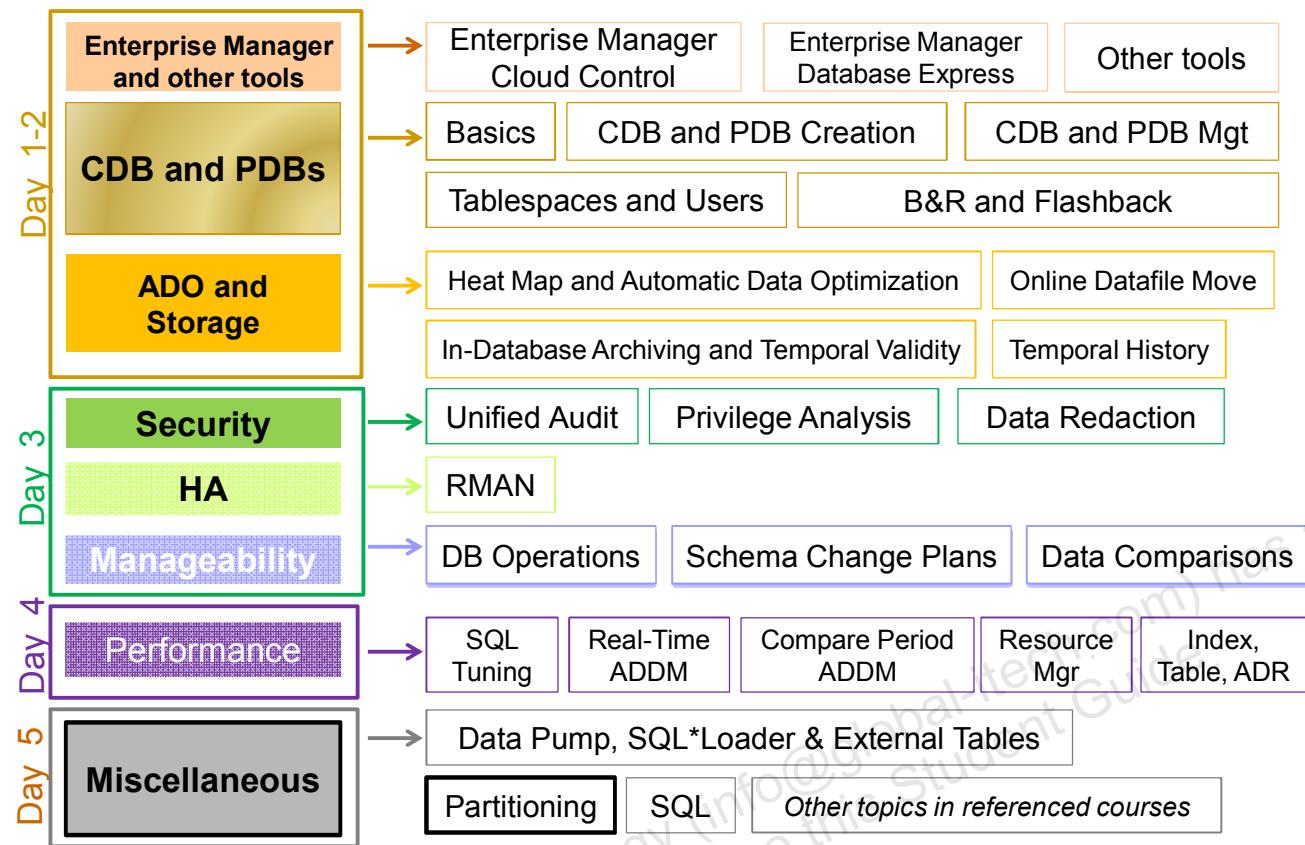
21

Partitioning Enhancements

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several **Partitioning** enhancements.

Objectives

After completing this lesson, you should be able to:

- Describe Online Move Partition
- Explain Interval Reference Partitioning
- Maintain multiple partitions in a single operation
- Describe partial local and global indexes
- Create and maintain partial local and global indexes
- Describe Asynchronous Global Index maintenance



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

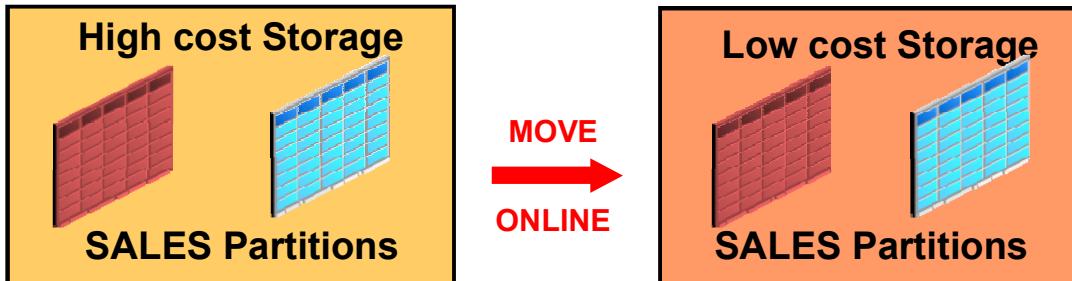
Note

For a complete understanding of partitioning enhancements and usage, refer to the following guides in the Oracle documentation:

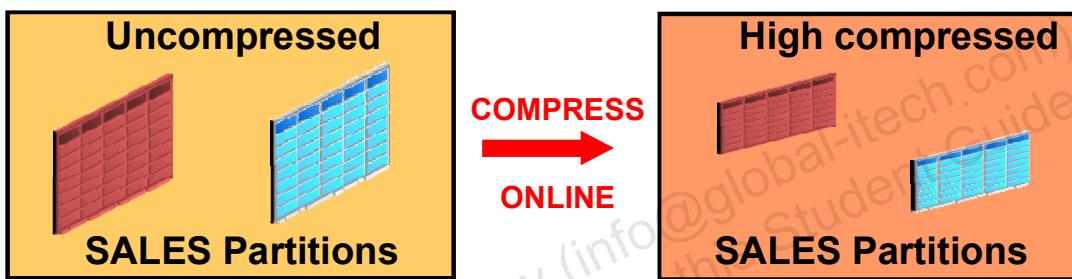
- *Oracle Database VLDB and Partitioning Guide 12c Release 1 (12.1)*
- *Oracle Database Data Warehousing Guide 12c Release 1 (12.1)*

Online Partition Operations

- **Move partitions** and subpartitions **ONLINE** to low-cost storage when they are infrequently accessed or updated.



- **Compress partitions** and subpartitions **ONLINE**.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In 11g, if you use conventional DML on a table partition compressed with basic compression or Hybrid Columnar Compression, then all inserted and updated rows are stored uncompressed or in a less-compressed format. To “pack” the compressed table partition so that these rows are compressed, use an `ALTER TABLE MOVE PARTITION` statement. This prevents any updates and loads until it completes.

12c Online Partition maintenance enhancements provide the capability to move table partitions or subpartitions online without preventing concurrent DML operations.

This can be used to:

- Move partitions and subpartitions from one kind of storage to another
- Move time based partitions and subpartitions to low cost storage after they become infrequently accessed, for example according to ILM policies configured
- Compress time based partitions and subpartitions according to ILM policies configured

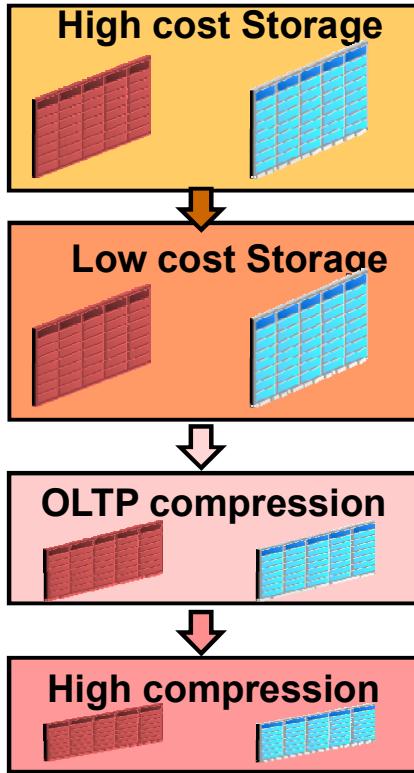
`ALTER TABLE MOVE (SUB) PARTITION` is made online to allow ILM policies to automatically move partitions or change partition compression properties in the background.

ILM can benefit from this new enhancement to move and compress table partitions according to ILM policies configured on the table or the partition.

When a partition meets the policy criterion set, the partition can be moved or compressed online under the scene.

Online Partition Operations: Benefits

SALES old partitions



- DML is allowed, not DDL.
- Move partitions to low-cost storage once infrequently accessed or updated.
- Global and local indexes maintained

```
SQL1> UPDATE sales SET c1=c1*2
2 WHERE DATE_P1='2001';
```

```
SQL2> ALTER TABLE sales
2 MOVE PARTITION sales_p1
3 TABLESPACE lowtbs
4 UPDATE INDEXES ONLINE;
```

- Compress partitions ONLINE.

```
SQL> ALTER TABLE sales
2 MOVE PARTITION sales_p1 COMPRESS
3 UPDATE INDEXES ONLINE;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

An ALTER TABLE MOVE PARTITION using ONLINE option allows DML operations to run uninterrupted on the partition that is being moved. The operation does not take an X DML lock on the (sub)partition being moved, but it prevents concurrent DDLs on the affected partitions, until the operation is completely done. The ALTER TABLE MOVE PARTITION ONLINE completes after all DMLs that started before the DDL statement complete.

In addition, global and local indexes are maintained during the move partition using the UPDATE INDEXES clause. A manual index rebuild is no longer required.

Online operation in the presence of domain indexes is not supported, when UPDATE INDEXES is specified. Online operation on IOTs is not supported.

Online Partition Operation: Compression

- COMPRESS BASIC

```
SQL> ALTER TABLE sales MOVE PARTITION sales_2000
  2      COMPRESS BASIC
  3      UPDATE INDEXES ONLINE;
```

- COMPRESS OLTP

```
SQL> ALTER TABLE sales MOVE PARTITION sales_2000
  2      COMPRESS FOR OLTP
  3      UPDATE INDEXES ONLINE;
```

- COMPRESS HCC

```
SQL> ALTER TABLE sales MOVE PARTITION sales_2000
  2      COMPRESS FOR QUERY
  3      UPDATE INDEXES ONLINE;
```

```
SQL> ALTER TABLE sales MOVE SUBPARTITION sales_2000_s1
  2      COMPRESS FOR ARCHIVE
  3      UPDATE INDEXES ONLINE;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

MOVE is typically the vehicle for compressing existing segments, therefore BASIC, OLTP and HCC compression types are supported.

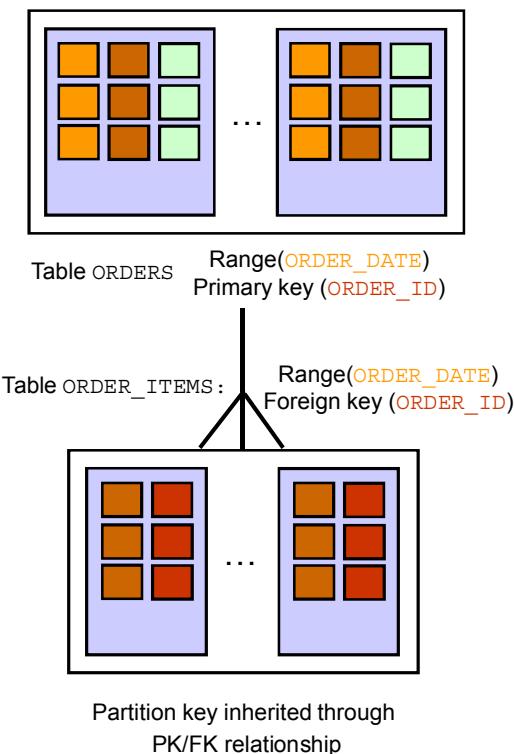
The first example shows how to compress, with BASIC level, the uncompressed partition sales_2000 online. Global and local indexes are maintained during the partition compression.

The second example shows how to compress, with OLTP level, the partition sales_2000 online. Global and local indexes are maintained during the partition compression.

The third and fourth examples shows how to compress, with a higher compression level, the partition sales_2000 or subpartition sales_2000_s1 online. Global and local indexes are maintained during the partition / subpartition compression.

- COMPRESS FOR QUERY LOW or HIGH provides a higher level of compression than OLTP compression. It works well when load performance is critical, frequent queries are run against the table, and no DML is expected. It enables HCC. (Default is HIGH.)
- COMPRESS FOR ARCHIVE LOW or HIGH compression provides the highest level of compression and works well for infrequently accessed data, mostly for read-only data. It enables HCC. (Default is LOW.)

Enhancements to Reference Partitioning



Three enhancements:

- Supports Interval Reference Partitioning
- Provides the CASCADE option for the TRUNCATE PARTITION operation
- Provides the CASCADE option for the EXCHANGE [SUB] PARTITION operation



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

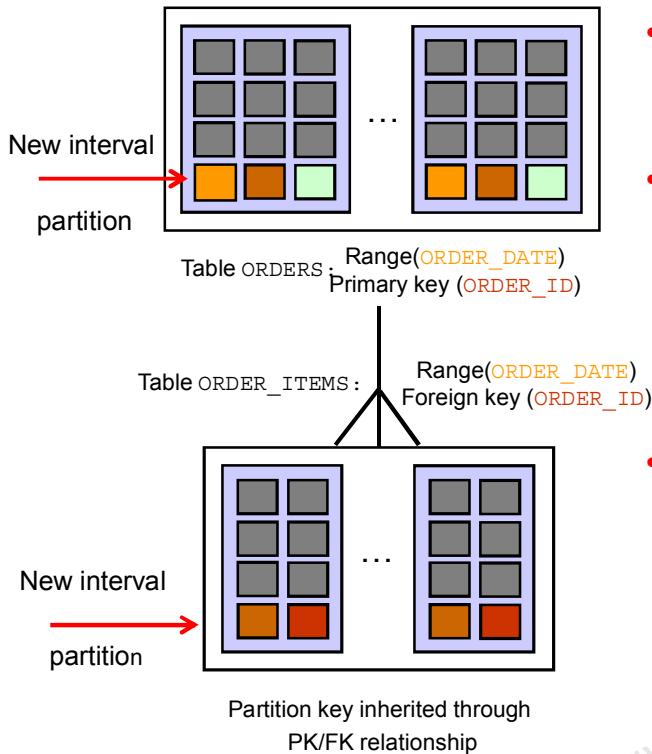
Reference partitioning was introduced in Oracle 11g. It equipartitions two tables related by referential constraints. When you partition a table by reference, partition maintenance operations subsequently performed on the parent table automatically cascade to the child table. Therefore, you cannot perform partition maintenance operations directly on a reference-partitioned table.

In Oracle Database 12c Release 1, reference partitioning is enhanced in three new ways:

- Interval Reference Partitioning allows reference partitioning with interval partitioned parent tables.
- Provide a CASCADE option for TRUNCATE PARTITION operations, which cascades the operation to reference-partitioned child tables.
- Provide a CASCADE option for EXCHANGE [SUB] PARTITION operations, which cascades the operation to reference-partitioned child tables.

Additionally, TRUNCATE TABLE is enhanced with a CASCADE option, which cascades the operation to child tables, (which do not have to be reference partitioned) based on referential constraints.

Interval Reference Partitioning



- Interval-partitioned tables can be used as parent tables for reference partitioning.
- Partitions in the reference-partitioned table corresponding to interval partitions in the parent table are created upon insertion into the reference partitioned table.
- Whenever an interval partition is created in a parent table, the partition name in the child table is inherited from the associated parent table.

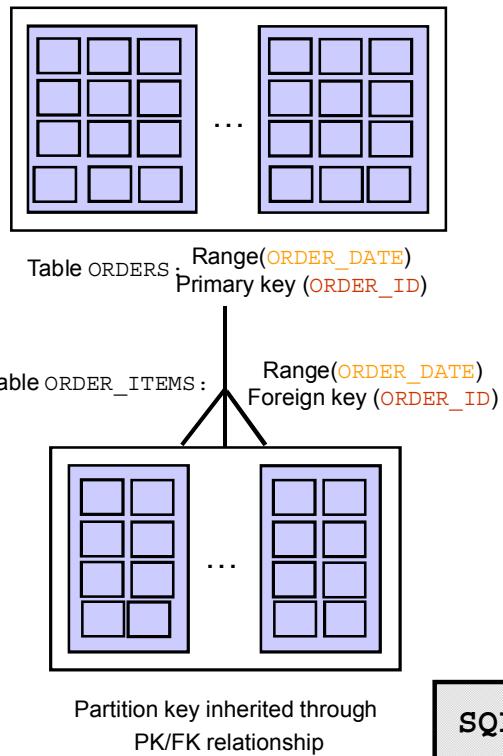
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Interval partitioned tables can be used as parent tables for reference partitioning. Partitions in the interval section of an interval reference-partitioned table (child table) are created when a row is inserted into the child table. Therefore, it is possible that an interval partition exists in the parent table, but it does not exist in the child table because no one inserted any row into that partition of the child table. When an interval partition is created in a parent table, the partition name in the child table is inherited from the associated parent table.

If the child table has a table-level default tablespace, it is used as the tablespace for the new interval partition. Otherwise, the tablespace is inherited from the parent table partition.

TRUNCATE TABLE CASCADE



If you specify the CASCADE option for TRUNCATE TABLE:

- Truncates all child tables that reference the table with an enabled ON DELETE CASCADE referential constraint.
- Succeeds if a parent and child are connected by multiple referential constraints and at least one of the constraints has ON DELETE CASCADE enabled.

```
SQL> TRUNCATE TABLE hr.orders CASCADE;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

As of Oracle Database 12c, you can add a CASCADE option for truncate operations to cascade the operation based on referential constraints. This affects TRUNCATE TABLE, ALTER TABLE TRUNCATE PARTITION, and ALTER TABLE TRUNCATE SUBPARTITION operations.

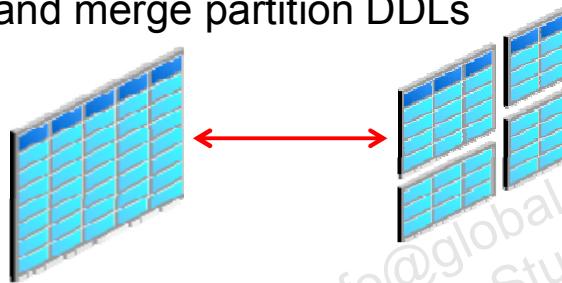
If you specify the CASCADE option for TRUNCATE TABLE, Oracle Database truncates all the child tables that reference the table with an enabled ON DELETE CASCADE referential constraint. This cascading action applies recursively to grandchildren, great-grandchildren, and so on. After determining the set of tables to be truncated based on the enabled ON DELETE CASCADE referential constraints, an error is raised if any table in this set is referenced through an enabled constraint from a child outside of the set. If a parent and child are connected by multiple referential constraints, a TRUNCATE TABLE CASCADE operation targeting the parent succeeds if at least one constraint has ON DELETE CASCADE.

The TRUNCATE can be targeted at any level in a reference-partitioned hierarchy and cascades to child tables starting from the targeted table. Privileges are not required on the child table. The CASCADE option is ignored if you specify it for a table that does not have reference-partitioned children. Any other options specified for the operation, such as DROP STORAGE or UPDATE INDEXES, apply to all the tables affected by the operation.

The CASCADE options are off by default.

Multi-Partition Maintenance Operations

- Add / truncate / drop partition allows you to add / truncate / drop multiple partitions in one single operation.
- Split partition and merge partitions operations allow you to roll out data into smaller partitions or to roll up data into larger partition
- Prior to Oracle Database 12c, Oracle allowed splitting into and merging of only two partitions using the ALTER TABLE split and merge partition DDLs



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Split partition and merge partitions operations allow you to roll out data into smaller partitions or to roll up data into a larger partition.

Prior to Oracle Database 12c, Oracle allowed splitting into and merging of only two partitions using the alter table split and merge partition DDLs. As a result, to split a partition into N partitions or to merge N partitions into one, $(N-1)$ DDLs must be issued. For instance, you may want to roll up data for the last year by merging all the monthly partitions. Because only two partitions can be merged at a time, it requires issuing 11 alter table merge partition DDLs. This is not only cumbersome, but it is also expensive due to multiple data reads and writes.

An alternative approach to merge multiple range partitions is to load the data from the N source partitions into a new non-partitioned table using the CREATE TABLE ... AS SELECT (CTAS) statement. Next, you drop the first $(N-1)$ source partitions and exchange the N -th source partition with the new table. The N -th source partition now contains data from the N partitions that were to be merged and may be renamed to the target partition name. This approach reduces data movement compared to issuing $(N-1)$ merge partition DDLs. For example, when rolling up monthly partitions for the last year, the customer will issue a CTAS, 11 drop partition and an exchange partition DDL to achieve the same result with lesser data movement.

Adding Multiple Partitions

- Add multiple new partitions with the ADD PARTITION clause of the ALTER TABLE statement.
- Local and global index operations are the same as when adding a single partition.
- Add multiple range partitions that are listed in ascending order of their upper bound values to the high end of a Range-partitioned or Composite Range-partitioned table.
- Add multiple list partitions to a table using new sets of partition values if the DEFAULT partition does not exist.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can add multiple new partitions with the ADD PARTITION clause of the ALTER TABLE statement. When adding multiple partitions, local and global index operations are the same as when adding a single partition. Note that both ADD PARTITION and ADD PARTITIONS are synonymous.

You can add multiple range partitions that are listed in ascending order of their upper bound values to the high end (after the last existing partition) of a Range-partitioned or Composite Range-partitioned table, provided the MAXVALUE partition is not defined. Similarly, you can add multiple list partitions to a table using new sets of partition values if the DEFAULT partition does not exist.

Creating a Range-Partitioned Table

```

CREATE TABLE sales
(
  prod_id NUMBER(6)
, cust_id NUMBER
, time_id DATE
, channel_id CHAR(1)
, promo_id NUMBER(6)
, quantity_sold NUMBER(3)
, amount_sold NUMBER(10,2)
)
PARTITION BY RANGE (time_id)
(PARTITION sales_q1_2012 VALUES LESS THAN (TO_DATE('01-APR-2012','dd-
  MON-YYYY')))
TABLESPACE tsa
,PARTITION sales_q2_2012 VALUES LESS THAN (TO_DATE('01-JUL-2012','dd-
  MON-YYYY'))
TABLESPACE tsb
,PARTITION sales_q3_2012 VALUES LESS THAN (TO_DATE('01-OCT-2012','dd-
  MON-YYYY'))
TABLESPACE tsc
,PARTITION sales_q4_2012 VALUES LESS THAN (TO_DATE('01-JAN-2013','dd-
  MON-YYYY'))
TABLESPACE tsd );

```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The slide example creates a SALES table with four partitions, one for each quarter of 2012.

This example shows how to create a range-partitioned table in Oracle Database 11g.

- Each partition is given a name:
 - sales_q1_2012
 - sales_q2_2012
 - sales_q3_2012
 - sales_q4_2012
- The time_id column is the partitioning column, whereas its value represents the partitioning key of a specific row.
- The VALUES LESS THAN clause determines the partition bound: Rows with partitioning key values that compare less than the ordered list of values specified by the clause are stored in the partition.
- Each partition is contained in a separate tablespace, tsa, tsb, tsc, and tsd.

For example, a row with time_id=17-MAR-2012 would be stored in partition sales_q1_2012.

Adding Multiple Partitions

- You can add multiple partitions using a single SQL statement by specifying the individual partitions as follows:

```
ALTER TABLE sales ADD
PARTITION sales_q1_2012 VALUES LESS THAN
    (TO_DATE('01-APR-2012', 'dd-MON-yyyy')) ,
PARTITION sales_q2_2012 VALUES LESS THAN
    (TO_DATE('01-JUL-2012', 'dd-MON-yyyy')) ,
PARTITION sales_q3_2012 VALUES LESS THAN
    (TO_DATE('01-OCT-2012', 'dd-MON-yyyy')) ,
PARTITION sales_q4_2012 VALUES LESS THAN
    (TO_DATE('01-JAN-2013', 'dd-MON-yyyy')) ;
;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 11g, you can add one partition to a partitioned table.

In Oracle Database 12c, you can add multiple partitions using a single statement by specifying the individual partitions. For example, as in the slide example, you add multiple partitions to the range-partitioned sales table created earlier named sales_q1_2012, sales_q2_2012, sales_q3_2012, and sales_q4_2012.

Truncating Multiple Partitions

- Truncate multiple partitions from a range- or list-partitioned table with the ALTER TABLE . . . TRUNCATE PARTITION statement.
- The corresponding partitions of local indexes are truncated in the operation.
- Global indexes must be rebuilt unless the UPDATE INDEXES clause is specified.

```
ALTER TABLE sales TRUNCATE PARTITIONS  
    sales_q1_2012, sales_q2_2012,  
    sales_q3_2012, sales_q4_2012;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Use the ALTER TABLE . . . TRUNCATE PARTITION statement to remove all rows from a table partition. Truncating a partition is similar to dropping a partition, except that the partition is emptied of its data, but not physically dropped.

The corresponding partitions of local indexes are truncated in the operation.

Global indexes must be rebuilt unless the UPDATE INDEXES clause is specified.

The slide example truncates four partitions in the range-partitioned sales table.

Dropping Multiple Partitions

- Remove multiple partitions or subpartitions from a range- or list-partitioned table with the following clauses of the **ALTER TABLE** statement:
 - **DROP PARTITION**
 - **DROP SUBPARTITION**

```
ALTER TABLE sales DROP PARTITIONS sales_q1_2012,  
                     sales_q2_2012,  
                     sales_q3_2012,  
                     sales_q4_2012;
```

- Local and global index operations are the same as when dropping a single partition.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can remove multiple partitions or subpartitions from a range or list partitioned table with the **DROP PARTITION** and **DROP SUBPARTITION** clauses of the **ALTER TABLE** statement. For example, the statement in the slide drops multiple partitions from the Range-partitioned table, **sales**. Note that you cannot drop all the partitions of a table.

When you drop multiple partitions, local and global index operations are the same as when you drop a single partition.

Splitting into Multiple Partitions

- Redistribute the contents of one partition into multiple partitions with the `SPLIT PARTITION` clause of the `ALTER TABLE` statement.
- Each new partition obtains a new segment and inherits all unspecified physical attributes from the current source partition.
- Use the extended split syntax to specify a list of new partition descriptions similar to the create partitioned table SQL statements, instead of using the `AT` or `VALUES` clauses.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can redistribute the contents of one partition into multiple partitions with the `SPLIT PARTITION` clause of the `ALTER TABLE` statement. When splitting multiple partitions, the segment associated with the current partition is preserved and new (empty) segments are created. Each new partition obtains a new segment and inherits all of the unspecified physical attributes from the current source partition.

You can use the extended split syntax to specify a list of new partition descriptions similar to the create partitioned table SQL statements, instead of using the `AT` or `VALUES` clauses. Additionally, the range or list values clause for the last new partition description is derived based on the high bound of the source partition and the bound values specified for the first ($N-1$) new partitions resulting from the split.

Splitting into Multiple Partitions: Examples

```
ALTER TABLE SPLIT PARTITION p0 INTO  
(PARTITION p01 VALUES LESS THAN (25),  
 PARTITION p02 VALUES LESS THAN (50),  
 PARTITION p03 VALUES LESS THAN (75),  
 PARTITION p04);
```

```
ALTER TABLE SPLIT PARTITION p0 INTO  
(PARTITION p01 VALUES LESS THAN (25),  
 PARTITION p02);
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The first slide example demonstrates splitting a partition `p0` into multiple partitions namely `p01`, `p02`, `p03`, and `p04`.

In the second slide example, partition `p02` has the high bound of the original partition `p0`.

Splitting into Multiple Partitions Rules

- Split a range partition into N partitions:
 - $(N-1)$ values of the partitioning key column must be specified within the range of the partition at which to split the partition.
- Split a list partition into N partitions:
 - $(N-1)$ lists of literal values must be specified.
 - Each list defines the first $(N-1)$ partitions into which rows with corresponding partitioning key values are inserted.
- When splitting a DEFAULT list partition or a MAXVALUE range partition into multiple partitions:
 - The first $(N-1)$ new partitions are created using the literal value lists or high bound values specified.
 - The N th new partition resulting from the split have the DEFAULT value or MAXVALUE.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To split a range partition into N partitions, $(N-1)$ values of the partitioning key column must be specified within the range of the partition at which to split the partition. The new non-inclusive upper bound values specified must be in ascending order. The high bound of N th new partition is assigned the value of the high bound of the partition being split. The names and physical attributes of the N new partitions resulting from the split can be optionally specified.

To split a list partition into N partitions, $(N-1)$ lists of literal values must be specified, each of which defines the first $(N-1)$ partitions into rows with corresponding partitioning key values are inserted. The remaining rows of the original partition are inserted into the N th new partition whose value list contains the remaining literal values from the original partition. No two value lists can contain the same partition value. The $(N-1)$ value lists that are specified cannot contain all of the partition values of the current partition because the N th new partition would be empty. Also, the new $(N-1)$ value lists cannot contain any partition values that do not exist for the current partition.

When splitting a DEFAULT list partition or a MAXVALUE range partition into multiple partitions, the first $(N-1)$ new partitions are created using the literal value lists or high bound values specified, whereas the N th new partition resulting from the split have the DEFAULT value or MAXVALUE. The SPLIT_TABLE_SUBPARTITION clause is extended similarly to allow split of a range or list subpartition into N new subpartitions.

Splitting into Multiple Partitions: Examples

```
ALTER TABLE sales SPLIT PARTITION sales_2012 INTO
(PARTITION sales_Q1_2012 VALUES LESS THAN (TO_DATE('01-
APR-2012','dd-MON-yyyy')) ,
PARTITION sales_Q2_2012 VALUES LESS THAN (TO_DATE('01-
JUL-2012','dd-MON-yyyy')) ,
PARTITION sales_Q3_2012 VALUES LESS THAN (TO_DATE('01-
OCT-2012','dd-MON-yyyy')) ,
PARTITION sales_Q4_2012);
```

```
ALTER TABLE list_customers SPLIT PARTITION Europe INTO
(PARTITION western-europe VALUES ('GERMANY', 'FRANCE') ,
PARTITION southern-europe VALUES ('ITALY') ,
PARTITION rest-europe);
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The first slide example splits the `sales_2012` partition of the partitioned table `sales` into four partitions corresponding to the quarters of the 2012 year. In this example, the partition `sales_2012` implicitly becomes the high bound of the split partition.

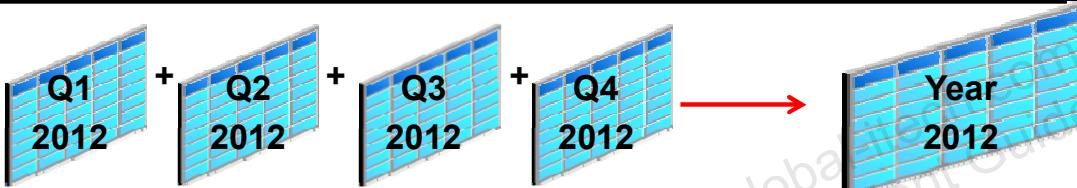
In the second slide example, the sample table `customers` partitioned by list, splits the partition `Europe` into three partitions: `western_europe`, `southern_europe`, and `rest_europe`.

Merging Multiple Range Partitions

- Merge the contents of two or more partitions or subpartitions into one new partition or subpartition.

```
ALTER TABLE sales
    MERGE PARTITIONS sales_q1_2012, sales_q2_2012,
                  sales_q3_2012, sales_q4_2012
    INTO PARTITION sales_2012;
```

```
ALTER TABLE sales
    MERGE PARTITIONS sales_q1_2012 TO sales_q4_2012
    INTO PARTITION sales_2012;
```



- The new partition inherits the partition upper bound of the highest of the original partitions

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 11g, you can merge the contents of two partitions into one partition.

In Oracle Database 12c, you can merge the contents of two or more partitions or subpartitions into one new partition or subpartition. You can then drop the original partitions or subpartitions with the MERGE PARTITIONS and MERGE SUBPARTITIONS clauses of the ALTER TABLE SQL statement as shown in the slide.

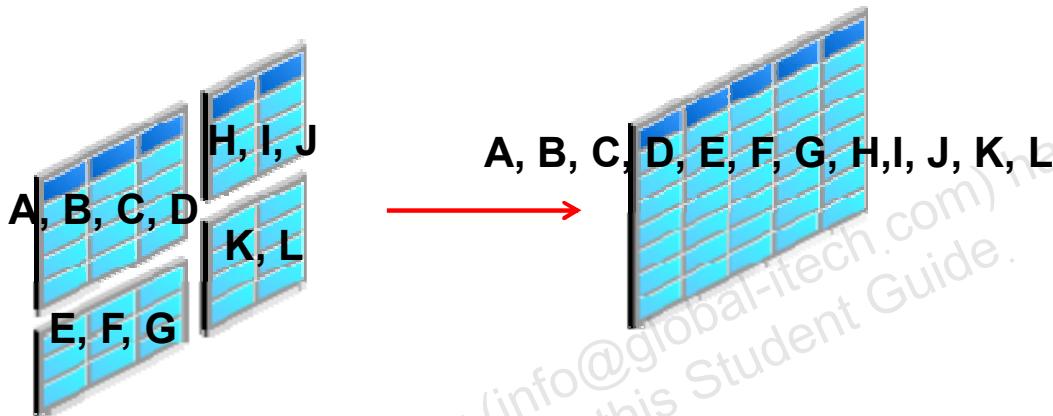
When merging multiple range partitions, the partitions must be adjacent and specified in the ascending order of their partition bound values. The new partition inherits the partition upper bound of the highest of the original partitions.

In the first slide example, you merge the four partitions of the sales range-partitioned table. These four partitions that correspond to the four quarters of the oldest year are merged into a single partition containing the entire sales data for the year.

If you are using range partitioning, you can use the second slide example to rewrite the first statement using the TO keyword.

Merging List and System Partitions

- When merging multiple list partitions, the resulting partition value list is the union of the set of partition value list of all of the partitions to be merged.
- A **DEFAULT** list partition merged with other list partitions results in a **DEFAULT** partition.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When merging multiple list partitions, the resulting partition value list are the union of the set of partition value list of all of the partitions to be merged. A **DEFAULT** list partition merged with other list partitions results in a **DEFAULT** partition.

Quiz

You can redistribute the contents of one partition into multiple partitions with the SPLIT PARTITION clause of the ALTER TABLE statement.

- a. True
- b. False

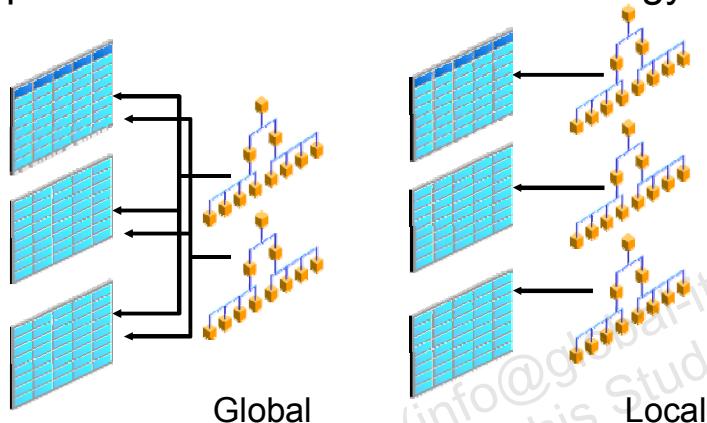


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Partitioned Indexes: Review

- Indexes can be partitioned like tables.
- A partitioned index can exist on a non-partitioned table.
- A global partitioned index uses a different partition strategy than that of the table.
- A local partitioned index follows the strategy of the table.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The same management benefits and performance improvement that can be achieved by partitioning tables is achieved by partitioning indexes. When an index is partitioned, each partition is a complete separate index. There is no master index to decide which index to use because this piece of information is inherent in the partition information. The syntax used to specify the partitioning of an index is very similar to that used to partition a table.

Global indexes can be defined on non-partitioned and partitioned tables and do not follow the partitioned table partitioning key. Local indexes can only be defined on partitioned tables. A local index has the same partitioning key as the table partitioning key. You can use only range and hash partitioning for global indexes.

Local indexes are automatically maintained; that is, changes made on the partitions on the table are automatically repeated on the local indexes; however, depending on the DDL, you may have to use the update indexes clause to maintain associated local index during partition maintenance operations.

A local index segment is always built for the equivalent data (table) segment. Therefore, for a composite partitioned tables, you have only local subpartitioned index segments and nothing on the logical partition top level.

Partial Indexes for Partitioned Tables

- Local and global indexes can be created on a **subset of the partitions** of a table, enabling more flexibility in index creation.
- This feature supports global indexes that index a certain subset of table partitions or subpartitions, and not others.
- This feature is supported by using a *default* table indexing property
- When a table is created or altered, a default indexing property can be specified for the table or its partitions.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Indexes typically occupy considerable space because each table in the database could have many indexes defined on it.

Global indexes index all table partitions and the index structure has no correlation with the table partitioning. Prior to Oracle Database 12c, Oracle did not provide the ability to index a subset of partitions and to exclude the others. There is no operation that a DBA may use prior to Oracle Database 12c to exclude non-active partitions from a global index.

In Oracle Database 12c, Partial Global Indexes save space and improve performance during loads and queries.

Local and global indexes can be created on a subset of the partitions of a table, enabling more flexibility in index creation.

This feature supports global indexes that include or index a certain subset of table partitions or subpartitions, and exclude the others. This operation is supported using a default table indexing property. When a table is created or altered, a default indexing property can be specified for the table or its partitions.

Partial Index Creation on a Table

- When an index is created as **PARTIAL** on a table:
 - Local indexes: An index partition is created usable if indexing is turned on for the table partition, and unusable otherwise
 - Global indexes: Include only those partitions for which indexing is turned on, and exclude the others
- **FULL** is the default if neither **FULL** or **PARTIAL** is specified at index creation.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When an index is created as **PARTIAL** on a table:

- Local indexes: An index partition is created **USABLE** if indexing is turned on for the table partition, and **UNUSABLE** otherwise.
- Global indexes: Include only those partitions for which indexing is turned on, and exclude the others.

You can override this behavior by specifying the **FULL** property at the index creation. **FULL** is the default if neither **FULL** or **PARTIAL** is specified. In this case, the index creation is similar to the behavior of Oracle Database releases prior to Oracle Database 12c. All local index partitions are created **USABLE** and a global index refers to all partitions of the table.

You cannot create a partial unique global index. A unique global index always has to be full.

Note

- In a local index, all keys in a particular index partition refer only to rows stored in a single underlying table partition. A partial local index is created by specifying the **LOCAL INDEXING PARTIAL** attribute.
- In a global index, the keys may refer to rows stored in multiple underlying table partitions or subpartitions. A partial global index is created by specifying the **GLOBAL INDEXING PARTIAL** attribute.

Specifying the INDEXING Clause at the Partition and Subpartition Levels

```

CREATE TABLE orders (
    order_id NUMBER(12),
    order_date DATE CONSTRAINT order_date_nn NOT NULL,
    ...
    CONSTRAINT order_total_min CHECK (order_total >= 0)
    INDEXING OFF
    PARTITION BY RANGE (ORDER_DATE)
    (PARTITION ord_p1 VALUES LESS THAN (TO_DATE('01-MAR-2010', 'DD-
        MON-YYYY')) INDEXING ON,
     PARTITION ord_p2 VALUES LESS THAN (TO_DATE('01-JUL-2010', 'DD-
        MON-YYYY')) INDEXING OFF,
     PARTITION ord_p3 VALUES LESS THAN (TO_DATE('01-OCT-2010', 'DD-
        MON-YYYY')) INDEXING ON,
     PARTITION ord_p4 VALUES LESS THAN (TO_DATE('01-MAR-2011', 'DD-
        MON-YYYY')),
     PARTITION ord_p5 VALUES LESS THAN (TO_DATE('01-MAR-2012', 'DD-
        MON-YYYY'))));

```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, the CREATE TABLE syntax is extended to support specification of the default indexing attributes clause INDEXING. You can use ALTER TABLE .. MODIFY DEFAULT ATTRIBUTES clause to change the default property of INDEXING ON/OFF. This changes the default for future partitions (and subpartitions). The INDEXING clause may also be specified at the partition and subpartition levels.

The slide example creates a table ORDERS with the following items:

- Partitions ORD_P1 and ORD_P3 are included in all partial global indexes.
- Partial local index partitions corresponding to the preceding two table partitions are created USABLE by default. The other local index partitions corresponding to table partitions with INDEXING OFF and table partitions with no INDEXING clause (ORD_P4 and ORD_P5), are created UNUSABLE.

Creating a Partial Local or Global Index

- An index, local or global, can be created to follow the table indexing properties by specifying the INDEXING PARTIAL clause.

```
CREATE INDEX orders_gidx_ordertotal  
    ON orders (order_total)  
    GLOBAL INDEXING PARTIAL;
```

- An index, local or global, can still be created to override the table indexing properties by specifying the INDEXING FULL clause.

```
CREATE INDEX orders_gidx_ordertotal  
    ON orders (order_total)  
    GLOBAL INDEXING FULL;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The ORDERS_GIDX_ORDERTOTAL global index in the slide example will be created to index only those table partitions that have INDEXING ON, and exclude the remaining partitions. Local indexes may also be created PARTIAL as discussed earlier, using the LOCAL clause.

Explain Plan: LOCAL INDEX ROWID

A partial local index is created on two partitions of the table on the columns ORDER_DATE.

```
SQL> EXPLAIN PLAN FOR SELECT order_mode, order_status FROM hr.tab_part1
  3 WHERE order_mode='direct';
SQL> select * from table(dbms_xplan.display) ;
```

| Id | Operation | Name | Pstart | Pstop |
|-------|-----------------------------------|----------------|-----------|----------|
| <hr/> | | | | |
| 0 | SELECT STATEMENT | | | |
| 1 | VIEW | VW_TE_2 | | |
| 2 | UNION-ALL | | | |
| 3 | CONCATENATION | | | |
| 4 | PARTITION RANGE SINGLE | | 1 | 1 |
| 5 | TABLE ACCESS FULL | TAB_PART1 | 1 | 1 |
| 6 | PARTITION RANGE SINGLE | | 3 | 3 |
| 7 | TABLE ACCESS BY LOCAL INDEX ROWID | | | |
| | | BATCHED | TAB_PART1 | 3 3 |
| 8 | INDEX RANGE SCAN | LIDX_ORDERDATE | 3 | 3 |
| 9 | PARTITION RANGE OR | | KEY (OR) | KEY (OR) |
| 10 | TABLE ACCESS FULL | TAB_PART1 | KEY (OR) | KEY (OR) |

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The LIDX_ORDERDATE partial local index is composed of two partitions indexing partitions 1 and 3 of the table that have INDEXING ON.

In this example, the partial local index is used to access rows in the partition 3 but not to access rows of partition 1. Most of the rows in partition 1 contain the value selected.

Explain Plan: GLOBAL INDEX ROWID

A partial global index is created on two partitions of the table on the column ORDER_MODE.

```
SQL> EXPLAIN PLAN FOR SELECT order_mode, order_status FROM hr.tab_part1
  3 WHERE order_mode='direct';
SQL> select * from table(dbms_xplan.display) ;
```

| Id | Operation | | Name | | Pstart | Pstop |
|----|------------------------------------|---------|----------------|----------|----------|----------|
| 0 | SELECT STATEMENT | | | | | |
| 1 | VIEW | | VW_TE_2 | | | |
| 2 | UNION-ALL | | | | | |
| 3 | TABLE ACCESS BY GLOBAL INDEX ROWID | BATCHED | TAB_PART1 | ROWID | ROWID | |
| 4 | INDEX RANGE SCAN | | GIDX_ORDERMODE | | | |
| 5 | PARTITION RANGE OR | | | | KEY (OR) | KEY (OR) |
| 6 | TABLE ACCESS FULL | | TAB_PART1 | KEY (OR) | KEY (OR) | |



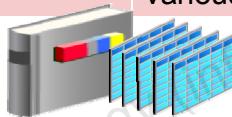
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The GIDX_ORDERMODE partial global index in the slide example indexes only those table partitions that have INDEXING ON, partitions 1 and 3 upon the five partitions of the table.

In this example, the partial global index is used to access rows in the table partitions 1 and 3 but not to access other rows in non-indexed partitions.

Affected Data Dictionary Views Overview

| Data Dictionary View / New columns | Description |
|---|---|
| DBA ALL USER_PART_TABLES New column DEF_INDEXING=ON/OFF | Display the object-level partitioning information for the partitioned tables. |
| DBA ALL USER_TAB_PARTITIONS New column INDEXING=ON/OFF | Display partition-level partitioning information, partition storage parameters, and partition statistics. |
| DBA ALL USER_TAB_SUBPARTITIONS New column INDEXING=ON/OFF | Display subpartition-level partitioning information, subpartition storage parameters, and subpartition statistics. |
| DBA ALL USER_INDEXES New column INDEXING =PARTIAL/FULL | Display indexes. |
| DBA ALL USER_IND_PARTITIONS Column STATUS=USABLE/UNUSABLE | Display for each index partition, the partition-level partitioning information, the storage parameters for the partition, and various partition statistics. |



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The following is a quick review of some of the affected views.

- The DBA_PART_TABLES view displays the object-level partitioning information for all partitioned tables in the database. The new column **DEF_INDEXING** has one of two values **ON** or **OFF**, specifying the default indexing **ON** or indexing **OFF** at the table level.
- The DBA_TAB_PARTITIONS view describes partition-level partitioning information, partition storage parameters, and partition statistics generated by the **DBMS_STATS** package for all partitions. The new column **INDEXING** has one of two values **ON** or **OFF**, specifying indexing **ON** or **OFF** at the partition level.
- The DBA_TAB_SUBPARTITIONS view describes, for each table subpartition, the subpartition name, the name of the table and partition to which it belongs, and its storage attributes. The new column **INDEXING** has one of two values **ON** or **OFF**, specifying indexing **ON** or **OFF** at the subpartition level.
- The DBA_INDEXES view describes indexes. To gather statistics for this view, you use the **DBMS_STATS** package. The new column **INDEXING** is **FULL** if the index is full or **PARTIAL** if the index is partial.
- The DBA_IND_PARTITIONS view displays, for each index partition, the partition-level partitioning information, the storage parameters for the partition, and various partition statistics generated by the **DBMS_STATS** package. The column **STATUS** is **USABLE** if the partial local index partition is used and **UNUSABLE** if not.

Asynchronous Global Index Maintenance

- DROP PARTITION and TRUNCATE PARTITION operations using the UPDATE INDEXES clause are optimized when maintaining global indexes:
 - Making the index maintenance a metadata-only operation
- Maintenance operations on indexes can be performed with the automatic scheduler job to clean up all global indexes:
 - SYS.PMO_DEFERRED_GIDX_MAINT_JOB
 - Executing DBMS_PART.CLEANUP_GID procedure

**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

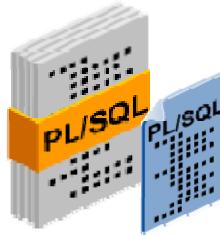
The partition maintenance operations DROP PARTITION and TRUNCATE PARTITION requesting global indexes maintenance are optimized making the index maintenance a metadata-only operation. This functionality enables maintenance of a list of data object numbers in metadata, where index entries corresponding to the drop and truncated objects that are invalid are ignored.

Maintenance operations on indexes can be performed with the automatic scheduler job SYS.PMO_DEFERRED_GIDX_MAINT_JOB to clean up all global indexes. This job is scheduled to run at 2:00 A.M. on a daily basis by default. You can run this job at any time using DBMS_SCHEDULER.RUN_JOB if you want to proactively clean up the indexes.

You can also modify the job to run with a different schedule based on your specific requirements. However, Oracle recommends that you do not drop the job.

The DBMS_PART Package

- The DBMS_PART package enables you to maintain and manage operations on partitioned objects.
- The new package contains the CLEANUP_GIDX procedure:
 - This procedure identifies and cleans up the global indexes to ensure efficiency in terms of storage and performance.



DBMS_PART.CLEANUP_GIDX

| SQL> desc dbms_part | | | | |
|------------------------|----------|--------|----------|--|
| PROCEDURE CLEANUP_GIDX | | | | |
| Argument Name | Type | In/Out | Default? | |
| SCHEMA_NAME_IN | VARCHAR2 | IN | DEFAULT | |
| TABLE_NAME_IN | VARCHAR2 | IN | DEFAULT | |

ORACLE

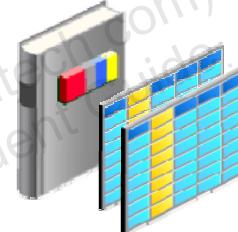
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The DBMS_PART package provides an interface for maintenance and management operations on partitioned objects. As a consequence of prior partition maintenance operations with asynchronous global index maintenance, global indexes can contain entries pointing to data segments that no longer exist. These stale index rows will not cause any correctness issues or corruptions during any operation on the table or index, whether these are queries, DMLs, DDLs, or analyze.

The DBMS_PART.CLEANUP_GIDX package procedure identifies and cleans up these global indexes to ensure efficiency in terms of storage and performance.

Global Index Maintenance Optimization During Partition Maintenance Operations

- *Partial Global Index Optimization:*
 - A new column ORPHANED_ENTRIES is added to the following data dictionary views:
 - DBA | ALL | USER_INDEXES
 - DBA | ALL | USER_IND_PARTITIONS
 - This column specifies whether or not a global index (partition) contains stale entries due to deferred index maintenance during one of the following operations:
 - DROP/TRUNCATE PARTITION
 - MODIFY PARTITION INDEXING OFF
 - The column will have one of three values:
 - YES, NO, or N/A



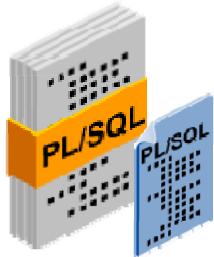
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Partial Global Index Optimization

- The column ORPHANED_ENTRIES is added to the dictionary views DBA_INDEXES and DBA_IND_PARTITIONS. This column specifies whether or not a global index (partition) contains stale entries due to deferred index maintenance during DROP/TRUNCATE PARTITION, or MODIFY PARTITION operations.
- The column can have one of three values:
 - YES: The index (partition) contains orphaned entries
 - NO: The index (partition) does not contain any orphaned entries
 - N/A: The property is not applicable; this is the case for local indexes, or indexes on non-partitioned tables.

Forcing an Index Cleanup: Additional Methods

You can also force cleanup of an index that needs maintenance using one of the following options:



`DBMS_PART.CLEANUP_GIDX`

```
SQL> ALTER INDEX ... REBUILD  
[PARTITION] ;
```

```
SQL> ALTER INDEX ... COALESCE  
[PARTITION] CLEANUP ;
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can also force a cleanup of an index needing maintenance using one of the following options:

- `DBMS_PART.CLEANUP_GIDX`: This PL/SQL package procedure gathers the list of global indexes in the system that may require cleanup and runs the operations necessary to restore the indexes to a clean state. This was covered earlier in this course.
- `ALTER INDEX REBUILD [PARTITION]`: This SQL statement rebuilds the entire index or index partition as was done prior to Oracle Database 12.1 releases; the resulting index (partition) does not contain any stale entries.
- `ALTER INDEX COALESCE [PARTITION] CLEANUP`: This SQL statement cleans up any orphaned entries in index blocks.

Quiz

The value of the new INDEXING column in the DBA_INDEXES view can be set to the following values (select all that apply):

- a. PARTIAL
- b. NO
- c. ON
- d. NA
- e. FULL



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a, e

Summary

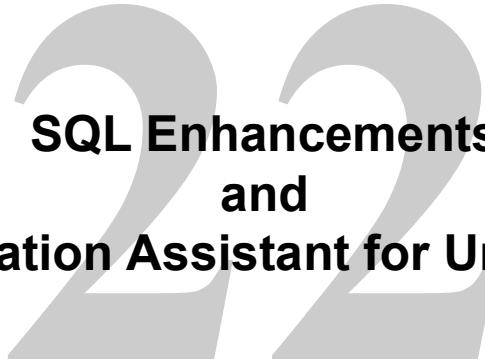
In this lesson, you should have learned how to:

- Describe Online Move Partition
- Explain Interval Reference Partitioning
- Maintain multiple partitions in a single operation
- Describe partial local and global Indexes
- Create and maintain partial local and global indexes
- Describe Asynchronous Global Index maintenance

Practice 21 Overview: Partitioning Enhancements

These practices cover the topic of use partial local and global indexing on partitioned tables.





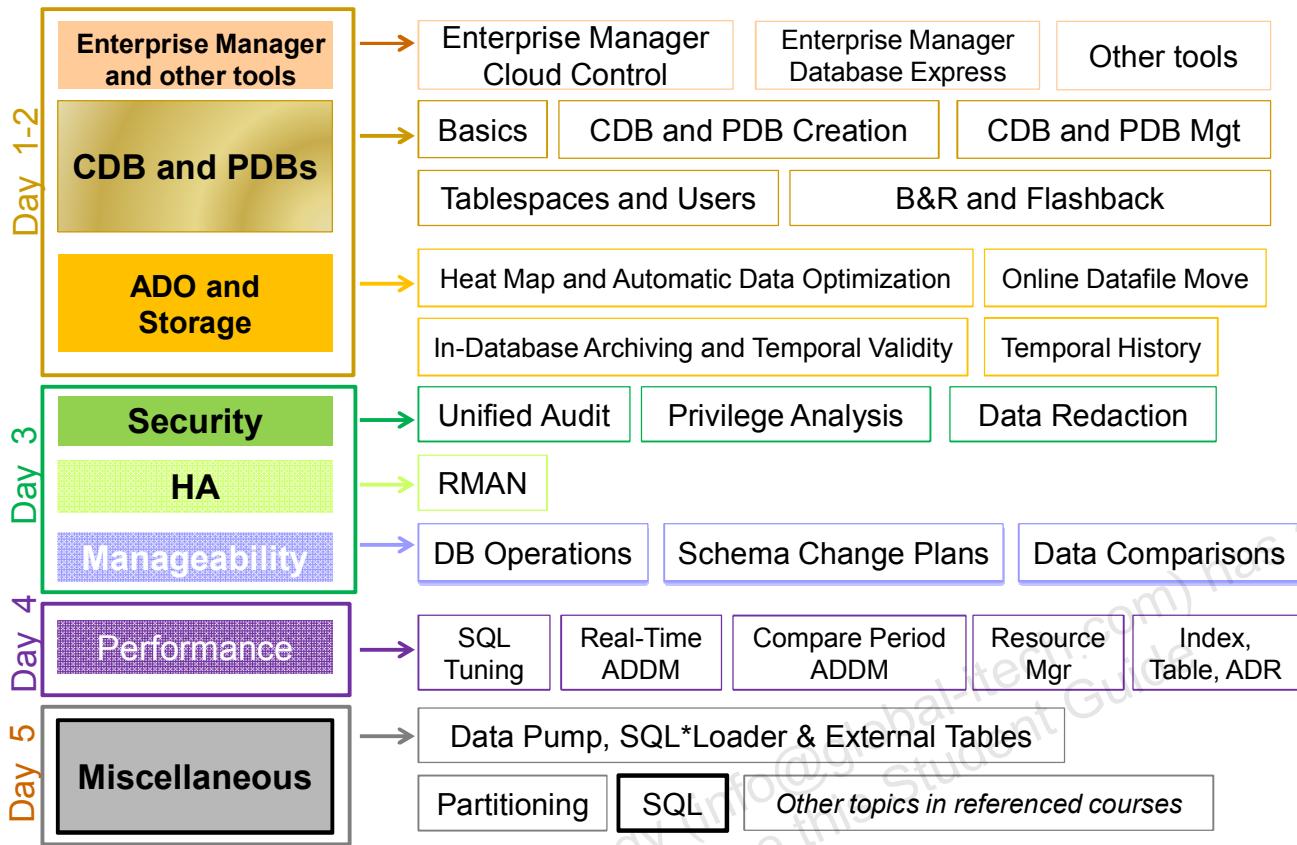
SQL Enhancements and Migration Assistant for Unicode



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c New and Enhanced Features



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are several **SQL** enhancements.

The **Migration Assistant for Unicode** replaces the Character Set Scanner.

Objectives

After completing this lesson, you should be able to:

- Enumerate the increase in the length limits for VARCHAR2, NVARCHAR2, and RAW data types in Oracle SQL
- Oracle Database Migration Assistant for Unicode 6.1
- Make SecureFiles the default storage mechanism for LOBs
- Use the SQL row-limiting clause in a query



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete understanding of the Oracle Data Redaction new feature usage, refer to the following guide in the Oracle documentation:

- *Oracle Database Reference 12c Release 1 (12.1) – “MAX_STRING_SIZE” Chapter*
- *Oracle Database SQL Language Reference 12c Release 1 (12.1) – “Data Types” Chapter*

Refer to other sources of information:

- *Oracle Database 12c New Features Demo Series* demonstrations under Oracle Learning Library:
 - *SQL Row_Limiting_Clause*
- *Oracle Database 12c: Installation and Upgrade* course

Increased Length Limits of Data Types

- The maximum size for the VARCHAR2, NVARCHAR2, and RAW data types is increased to 32767 bytes.

VARCHAR2 and NVARCHAR2
columns with a declared column length of 4000 bytes or less and RAW columns with a declared column length of 2000 bytes or less are stored inline.

VARCHAR2 and NVARCHAR2
columns with a declared column length of greater than 4000 bytes and RAW columns with a declared column length of greater than 2000 bytes are called extended character data type columns and are stored out-of-line.

- MAX_STRING_SIZE controls the maximum size of the extended data types in SQL.

```
MAX_STRING_SIZE = { STANDARD | EXTENDED }
```

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, you can specify a maximum size of 32767 bytes for the VARCHAR2, NVARCHAR2, and RAW data types. This enables users to store longer strings in the database. Prior to this release, the maximum size was 4000 bytes for the VARCHAR2 and NVARCHAR2 data types and 2000 bytes for the RAW data type. The declared length of the VARCHAR2, NVARCHAR2, or RAW column affects how the column is internally stored.

- VARCHAR2 and NVARCHAR2 columns with a declared column length of 4000 bytes or less and RAW columns with a declared column length of 2000 bytes or less are stored inline.
- VARCHAR2 and NVARCHAR2 columns with a declared column length of greater than 4000 bytes and RAW columns with a declared column length of greater than 2000 bytes are called “extended character data type columns” and are stored out-of-line.

MAX_STRING_SIZE controls the maximum size of the extended data types in SQL:

- STANDARD means the length limit of the data types used prior to Oracle 12c.
- EXTENDED means the 32767 bytes limit in Oracle Database 12c.

Extended character data types have the following restrictions:

- Not supported in clustered tables and index-organized tables
- No intrapartition parallel DDL, UPDATE, and DELETE DML
- No intrapartition parallel direct-path inserts on tables stored in a tablespace managed with Automatic Segment Space Management (ASSM).

Configuring Database for Extended Data Type

1. Shut down the database instance.
2. Restart the database in UPGRADE mode.
3. Change the setting of MAX_STRING_SIZE to EXTENDED ;

```
SQL> ALTER SYSTEM SET MAX_STRING_SIZE = EXTENDED;
```

4. Run the \$ORACLE_HOME/rdbms/admin/utl32k.sql script as SYSDBA.
5. Restart the database instance.

Note:

- You cannot change the value from EXTENDED to STANDARD.
- In a RAC environment, shut down all instances.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You cannot create a table with extended character data type columns until you configure the database with the instance parameter MAX_STRING_SIZE set to EXTENDED.

Follow the steps described on the slide.

This value is also enforced to be the same on all RAC nodes in a RAC environment.

Using VARCHAR2, NVARCHAR2, and RAW Data Types

- You create a table with extended character data type columns:

```
SQL> CREATE TABLE long_varchar(id NUMBER,vc VARCHAR2(32767));
```

- You can modify the size of existing VARCHAR2, NVARCHAR2, and RAW columns:

```
SQL> ALTER TABLE t MODIFY (varchar_column VARCHAR2(32767));
```

- You can add a 32k column to an existing table:

```
SQL> ALTER TABLE t ADD (long_varchar_column VARCHAR2(12345));
```

- Data Pump export, import and SQL*Loader can use extended Data Types
- Indexes prevent data type extension on existing columns.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can create a table with extended character data type columns as described on the slide.

You can modify the size of existing VARCHAR2, NVARCHAR2, and RAW columns with the ALTER TABLE MODIFY (column...) statement. In this case, Oracle Database performs an in-place length extension and does not migrate the inline storage to external LOB storage.

Note: Oracle recommends that you do not excessively increase the size of an existing VARCHAR2 column beyond 4000 bytes:

- Row chaining may occur.
- Any data stored inline has to be read in its entirety, whether or not a column is selected. Extended character data type columns stored inline can therefore negatively impact the performance.
- To migrate to the new out-of-line storage of extended character data type columns, you have to **recreate** the tables. Otherwise, the inline storage of the column is preserved during any type of table reorganization such as ALTER TABLE MOVE.

You can add a 32k column to an existing table via ALTER TABLE ADD DDL if the table satisfies the conditions required for 32k types, as described in the earlier CREATE TABLE section.

Data Pump export, import and SQL*Loader can use extended Data Types.

Existing indexes prevent data type extension on existing columns. You must drop indexes first then modify the column with the new extended value, and last recreate the indexes.

Database Migration Assistant for Unicode

Unicode Update: latest version is Unicode Standard 6.1

- Is an industry standard that allows text and symbols from all languages to be consistently represented and manipulated by computers

The Database Migration Assistant for Unicode 6.1 (DMU):

- Unique next-generation migration tool that provides a streamlined end-to-end solution for migrating databases from standard character sets to Unicode
- CSSCAN and CSALTER utilities removed from the database installation and unsupported
- Supports migration of all Oracle data types that contains textual data and deals with database objects
- Significantly reduces migration down time
- Located in \$ORACLE_HOME/dmu



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

DMU is a unique next-generation migration tool that provides a streamlined end-to-end solution for migrating databases from standard character sets to Unicode. The standard CSSCAN and CSALTER utilities will be removed from the database installation and desupported. The DMU supports the migration of almost all Oracle data types that may directly or indirectly contain textual data and deals with database objects, such as materialized views, indexes, constraints, and triggers, which are affected by conversion of tables so that they are properly synced up after the migration. It significantly reduces migration downtime. It is not compatible with CDBs.

Features:

- Guided end-to-end migration workflow
- Intuitive graphical user interface
- Advanced data analysis and cleansing tools
- Validation mode to check data integrity for compliance with the Unicode Standard

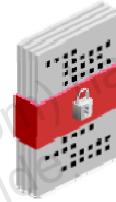
Benefits:

- Alleviates costly manual workload through automated migration tasks
- Simplifies data preparation process and prevents data loss
- Robust error-handling and failure recovery
- Health check on data integrity in databases encoded with the Unicode Standard

SecureFiles

SecureFiles:

- SecureFiles are made default storage mechanisms for large objects (LOBs).
- The default value for `DB_SECUREFILE` is PREFERRED when the `COMPATIBLE` `init.ora` parameter is set to 12.0.0.0 or higher.
- To change this behavior, set the value to PERMITTED.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Beginning with Oracle Database 12c, SecureFiles are now the default for large object (LOB) storage. If no storage type is explicitly specified, new LOB columns use SecureFiles LOB storage.

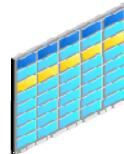
When the `COMPATIBLE` instance parameter is set to 12.0.0.0 or higher, the default value for `DB_SECUREFILE` is PREFERRED. To change this behavior, set the value to PERMITTED. SecureFiles provide optimal performance for storing unstructured data in the database. Making SecureFiles the default for unstructured data helps ensure that the database delivers the best possible performance when managing unstructured data.

SQL Row-Limiting Clause

Use a row-limiting clause to limit the rows returned by a query.

- Specify the number of rows to return with the `FETCH FIRST/NEXT` keywords.
- Specify the percentage of rows to return with the `PERCENT` keyword.
- Use the `OFFSET` keyword to specify that the returned rows begin with a row after the first row of the full result set.

Queries that order data and limit row output are widely used and are often referred to as `Top-N` queries.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c Release 1, SQL `SELECT` syntax has been enhanced to allow a row limiting clause, which limits the number of rows that are returned in the result set.

Limiting the number of rows returned can be valuable for reporting, analysis, data browsing, and other tasks. Queries that order data and then limit row output are widely used and are often referred to as `Top-N` queries.

You can specify the number of rows or percentage of rows to return with the `FETCH FIRST/NEXT` keywords. You can use the `OFFSET` keyword to specify that the returned rows begin with a row after the first row of the full result set.

SQL Row-Limiting Clause: Examples

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
FETCH FIRST 5 ROWS ONLY;
```

| Script Output X | |
|-----------------------------|-----------|
| Task completed in 0 seconds | |
| EMPLOYEE_ID FIRST_NAME | |
| 100 | Steven |
| 101 | Neena |
| 102 | Lex |
| 103 | Alexander |
| 104 | Bruce |

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

| Script Output X | |
|-----------------------------|--------|
| Task completed in 0 seconds | |
| EMPLOYEE_ID FIRST_NAME | |
| 105 | David |
| 106 | Valli |
| 107 | Diana |
| 108 | Nancy |
| 109 | Daniel |

Returns the 5 employees with the lowest employee_id

Returns the 5 employees with the next set of lowest employee_id

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You specify the row limiting clause in the SQL SELECT statement by placing it after the ORDER BY clause. Note that an ORDER BY clause is not required.

- **OFFSET:** Use this clause to specify the number of rows to skip before row limiting begins. The value for offset must be a number. If you specify a negative number, offset is treated as 0. If you specify NULL or a number greater than or equal to the number of rows that are returned by the query, 0 rows are returned.
- **ROW | ROWS:** Use these keywords interchangeably. They are provided for semantic clarity.
- **FETCH:** Use this clause to specify the number of rows or percentage of rows to return.
 - **FIRST | NEXT:** Use these keywords interchangeably. They are provided for semantic clarity.
 - **row_count | percent PERCENT:** Use row_count to specify the number of rows to return. Use percent PERCENT to specify the percentage of the total number of selected rows to return. The value for percent must be a number.

The first code example returns the five employees with the lowest employee_id.

The second code example returns the five employees with the next set of lowest employee_id.

Quiz

To use Extended Character Data types, you must set MAX_STRING_SIZE to EXTENDED.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a

Quiz

You can limit the number of rows returned by a query using:

- a. A SQL row-limiting clause in the query
- b. The session parameter `SQL_ROW_LIMITING`



Summary

In this lesson, you should have learned how to:

- Enumerate the increase in the length limits for VARCHAR2, NVARCHAR2, and RAW data types in Oracle SQL
- Oracle Database Migration Assistant for Unicode 6.1
- Making the SecureFiles the default storages mechanism for LOBs
- Use the SQL row-limiting clause in a query

Practice 22 Overview: SQL Enhancements

These practices cover the following topics:

- Creating a table with a extended data type column (*optional*)
- Querying a table using a SQL row-limiting clause (*optional*)

New and Enhanced Features in Other Courses

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Further Information

For more information about topics that are not covered in this course, refer to the following:

- Other Oracle University Oracle Database 12c ILT courses
- Oracle Database 12c: New Features Self Studies
 - A comprehensive series of self-paced online courses covering all new features in detail
 - Demonstrations for all topics covered in self-paced online courses:
<http://www.oracle.com/goto/oll>
- Oracle By Example series: Oracle Database 12c
 - <http://www.oracle.com/technology/obe/demos/admin/demos.html>
 - <http://www.oracle.com/technology/obe/start/index.html>
- Oracle OpenWorld events
 - <http://www.oracle.com/openworld/index.html>



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

For more information about topics that are not covered in this course, refer to the following:

- Oracle University Oracle Database 12c instructor-led courses
- Oracle Database 12c: New Features self-paced online courses
- Oracle By Example series: Oracle Database 12c
- Oracle OpenWorld events

Suggested Oracle University ILT Courses

- Enterprise Manager Cloud Control
 - *Using Oracle Enterprise Manager Cloud Control 12c*
 - *Oracle Enterprise Manager Cloud Control 12c: Install and Upgrade*
 - *Oracle Enterprise Manager Cloud Control 12c: Advanced Configuration*
 - *Oracle Enterprise Manager Cloud Control 12c: Cloud Management workshop*
- Installation and Upgrade
 - *Oracle Database 12c: Install and Upgrade workshop*



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Additional Courses

To obtain more information about the key cloud computing technologies used by the Oracle products and other new Oracle Database features, follow additional training from Oracle University.

Suggested Oracle University ILT Courses

- Real Application Cluster
 - *Oracle Database 12c: High Availability New Features Ed 1.*
 - *Oracle Database 12c: Manage Clusterware workshop*
 - *Oracle Database 12c: Manage ASM Workshop*
 - *Oracle Database 12c: RAC Administration workshop*
- Security
 - *Oracle Database 12c: Security*
- Performance
 - *Oracle Database 12c: Performance Tuning workshop*
- Exadata
 - *Exadata Database Machine Administration workshop Ed1*



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Suggested Additional Courses

To obtain more information about the key cloud computing technologies used by the Oracle products and other new Oracle Database features, follow additional training from Oracle University.

New Processes, Views, Parameters, Packages, and Privileges



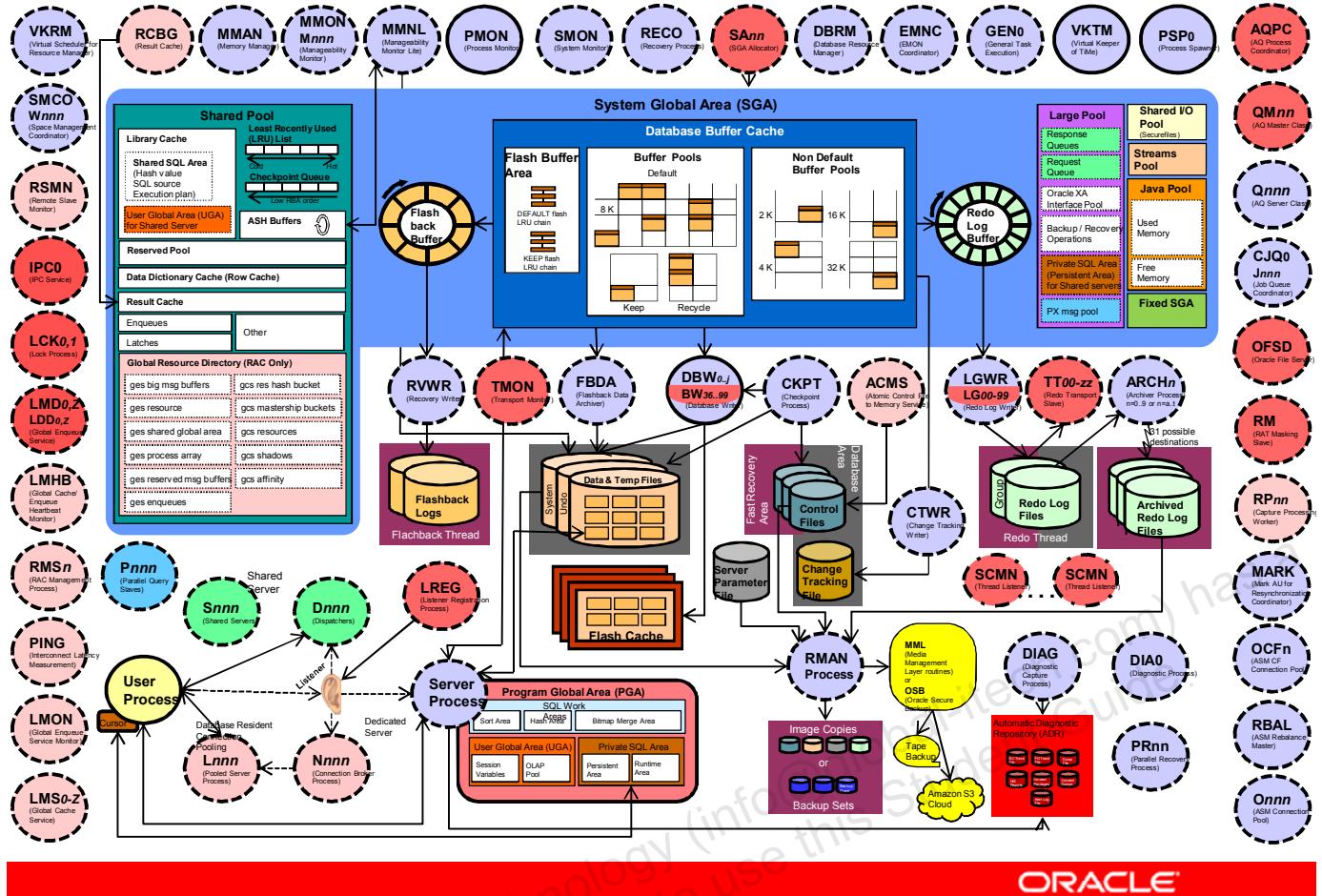
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Note

For a complete list of views, parameters, packages, and privileges on Oracle Database 12c new features, refer to the following guide in the Oracle documentation:

- *Oracle Database Reference 12c Release 1 (12.1)*
- *Oracle Database PL/SQL Packages and Types Reference 12c Release 1 (12.1)*
- *Oracle Database Security Guide 12c Release 1 (12.1)*



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

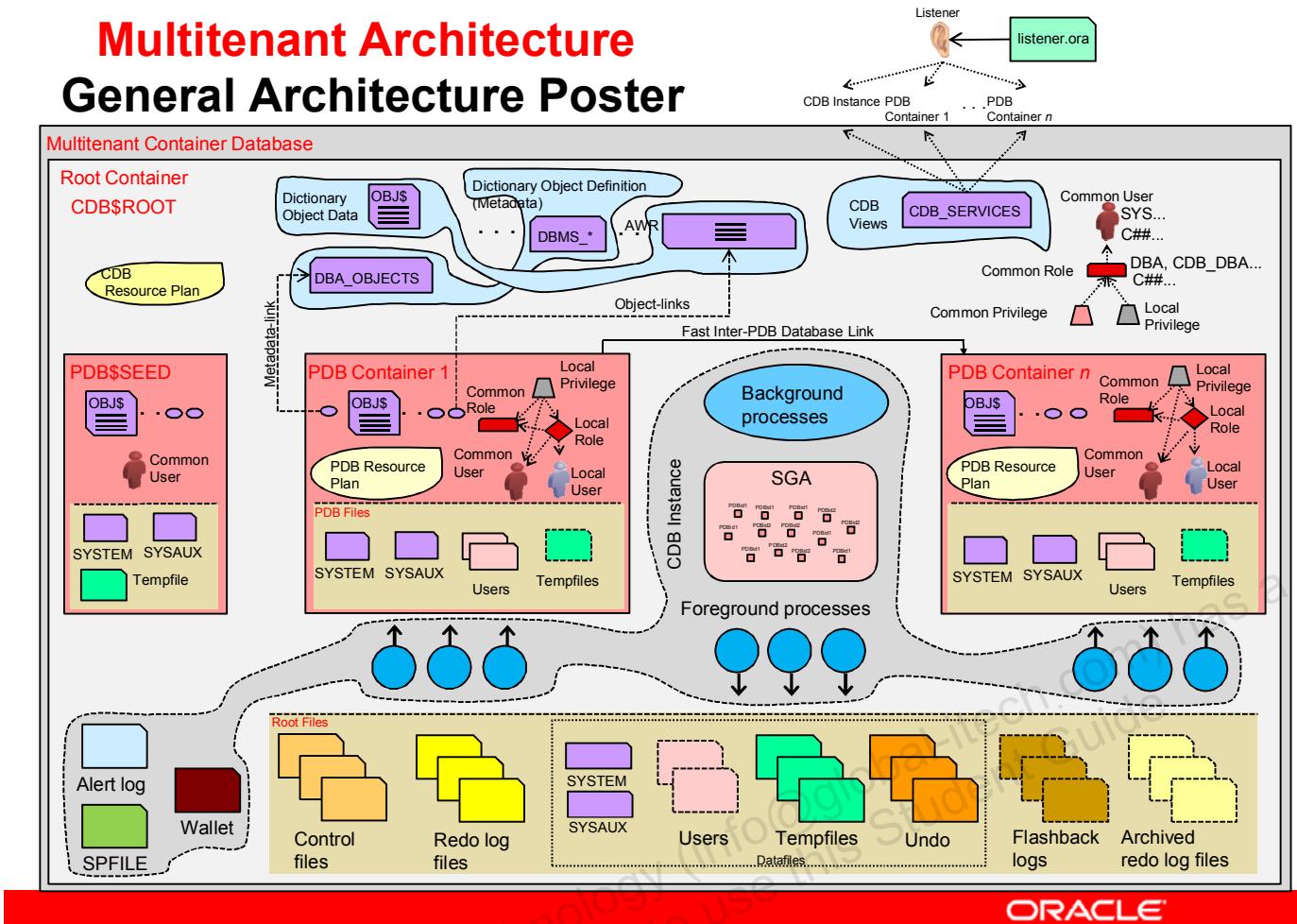
Caption:

1. Circle elements represent Oracle processes. If they are surrounded by dotted lines (equal dotted elements) it means they can run either as threads or OS processes. If they are surrounded by a solid line it means they can only run as OS processes. The SCMN case is an exception. When using the Multi-Process Multi-Threaded architecture, each OS process, running more than one Oracle Process, also runs a special thread called SCMN that is basically an internal listener thread. All thread creation is routed through this thread.
2. Darker circle elements are new elements for DB 12c.
3. The two main different color nuances for circle elements are to make the distinction between RAC and non-RAC processes.
4. Files are represented by cylinders.
5. Storage location for those files are divided into three main areas: Fast Recover Area, Database Area, and Automatic Diagnostic Repository. Areas are designated by background rectangles using three different colors. Exception is the server parameter file. If you find a file type part of two areas it means that some corresponding files can be in both areas.
6. Inside one circle element you may see two names. This is to indicate the second (smaller letters) is a slave of the first.

Multitenant Architecture

General Architecture Poster

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

CDB and PDB

View

- CDB_xxx: All of the objects in the CDB across all PDBs
- DBA_xxx: All of the objects in a container or PDB
- CDB_pdbs: All PDBS within CDB
- CDB_tablespaces: All tablespaces within CDB
- CDB_users: All users within CDB (common and local)
- V\$PDBS: Displays information about PDBs associated with the current instance
- V\$CONTAINERS : Displays information about PDBs and the root associated with the current instance
- PDB_PLUG_IN_VIOLATIONS : Displays information about PDB violations after compatibility check with CDB
- RC_PDBS: Recovery catalog view about PDB backups

CDB and PDB

Parameter

- `ENABLE_PLUGGABLE_DATABASE=true`: required to create a CDB at CDB instance startup
- `PDB_FILE_NAME_CONVERT`: maps names of existing files to new file names when processing a `CREATE PLUGGABLE DATABASE` statement
- `CDB_COMPATIBLE=true`: enables you to get behavior similar to a non-CDB

Package

- `DBMS_PDB.DESCRIBE`
- `DBMS_PDB.CHECK_PLUG_COMPATIBILITY`



ILM: Heat Map and ADO

View

- DBA_HEAT_MAP_SEG_HISTOGRAM
- DBA_HEAT_MAP_SEGMENT
- V\$HEAT_MAP_SEGMENT
- DBA_ILMOBJECTS
- DBA_ILMPOLICIES, DBA_ILMDATAMOVEMENTPOLICIES
- DBA_ILMTASKS, DBA_ILMEVALUATIONDETAILS
- DBA_ILMRESULTS

Parameter

- HEAT_MAP=ON activates activity tracking and statistics collection



ILM: Heat Map and ADO

Package

- DBMS_HEAT_MAP
 - BLOCK_HEAT_MAP
 - EXTENT_HEAT_MAP
- DBMS_ILM
 - EXECUTE_ILM
 - PREVIEW_ILM, ADD_TO_ILM, REMOVE_FROM_ILM
 - EXECUTE_ILM_TASK
 - STOP_ILM
- DBMS_ILM_ADMIN
 - CUSTOMIZE
 - DISABLE_ILM, ENABLE_ILM
 - CLEAR_HEAT_MAP_ALL, CLEAR_HEAT_MAP_TABLE
 - SET_HEAT_MAP_START
 - SET_HEAT_MAP_ALL, SET_HEAT_MAP_TABLE



In-Database Archiving and Temporal Validity

New column

- ORA_ARCHIVE_STATE in application tables

Package

- DBMS_FLASHBACK_ARCHIVE
 - ENABLE_AT_VALID_TIME

Security: Auditing

View

- UNIFIED_AUDIT_TRAIL
- AUDIT_UNIFIED_POLICIES
- AUDIT_UNIFIED_ENABLED_POLICIES

New columns in UNIFIED_AUDIT_TRAIL

- FGA_POLICY_NAME
- DP_xxx (Data Pump operations)
- RMAN_xxx
- OLS_xxx
- DV_xxx (Database Vault operations)
- XS_xxx (Real Application Security operations)

Package

- DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL



Security: Privilege Analysis

View

- DBA_USED_PRIVS
- DBA_USED_SYSPRIVS
- DBA_USED_OBJPRIVS
- DBA_USED_PUBPRIVS
- DBA_USED_OBJPRIVS_PATH
- DBA_USED_SYSPRIVS_PATH
- DBA_UNUSED_PRIVS
- DBA_UNUSED_OBJPRIVS
- DBA_UNUSED_SYSPRIVS
- DBA_UNUSED_OBJPRIVS_PATH
- DBA_UNUSED_SYSPRIVS_PATH
- DBA_PRIV_CAPTURES

Security: Privilege Analysis and New Privileges

Package

- DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE
- DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE
- DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE
- DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT
- DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE

Privilege

- SYSBACKUP
- SYSDG
- SYSKM
- PURGE DBA_RECYCLEBIN

Security: Oracle Data Redaction

View

- REDACTION_POLICIES
- REDACTION_COLUMNS
- REDACTION_VALUES_FOR_TYPE_FULL

Package

- DBMS_REDACT.ADD_POLICY
- DBMS_REDACT.ALTER_POLICY
- DBMS_REDACT.DROP_POLICY
- DBMS_REDACT.ENABLE_POLICY
- DBMS_REDACT.DISABLE_POLICY
- DBMS_REDACT.UPDATE_FULL_REDACTION_VALUES

Privilege

- EXEMPT REDACTION POLICY
- EXEMPT DDL REDACTION POLICY
- EXEMPT DML REDACTION POLICY



HA: Flashback Data Archive

Package

- DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL
- DBMS_FLASHBACK_ARCHIVE.GET_SYS_CONTEXT

Manageability: Database Operations

New Column

- DBOP_NAME
- DBOP_EXEC_ID in
 - V\$SQL_MONITOR
 - V\$ACTIVE_SESSION_HISTORY
 - CDB_HIST_ACTIVE_SESS_HISTORY
 - DBA_HIST_ACTIVE_SESS_HISTORY

Parameter

- STATISTICS_LEVEL=TYPICAL
- CONTROL_MANAGEMENT_PACK_ACCESS=DIAGNOSTIC+TUNING

Manageability: Database Operations

Package

- DBMS_SQL_MONITOR.BEGIN_OPERATION
- DBMS_SQL_MONITOR.END_OPERATION
- DBMS_SQL_MONITOR.REPORT_SQL_MONITOR
- DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST
- DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_XML
- DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_LIST_XML

Manageability: Data Comparisons

Package

- DBMS_COMPARISON

Performance: SQL Tuning

Package

- DBMS_SPM.REPORT_AUTO_EVOLVE_TASK
- DBMS_SPM.CREATE_EVOLVE_TASK
- DBMS_SPM.EXECUTE_EVOLVE_TASK
- DBMS_SPM.REPORT_EVOLVE_TASK
- DBMS_SPM.IMPLEMENT_EVOLVE_TASK
- DBMS_STATS.SEED_COL_USAGE
- DBMS_STATS.REPORT_COL_USAGE

Parameter

- OPTIMIZER_ADAPTIVE_REPORTING_ONLY

New Column

- IS_RESOLVED_DYNAMIC_PLAN, IS_REOPTIMIZABLE in V\$SQL

View

- DBA_SQL_PLAN_DIR_OBJECTS



Performance: ADDM

Package DBMS_ADDM:

New functions

- REAL_TIME_ADDM_REPORT
- COMPARE_INSTANCES
- COMPARE_DATABASES
- COMPARE_CAPTURE_REPLY_REPORT
- COMPARE_REPLY_REPLY_REPORT

Performance: Resource Manager

View

- DBA_CDB_RSRC_PLAN_DIRECTIVES

Package DBMS_RESOURCE_MANAGER: new procedures

- CREATE_CDB_PLAN
- UPDATE_CDB_PLAN
- DELETE_CDB_PLAN
- CREATE_CDB_PLAN_DIRECTIVE
- UPDATE_CDB_PLAN_DIRECTIVE
- DELETE_CDB_PLAN_DIRECTIVE
- UPDATE_CDB_DEFAULT_DIRECTIVE
- UPDATE_CDB_AUTOTASK_DIRECTIVE

Performance: Multi-Process Multi-Threaded

Parameter

- THREADED_EXECUTION=TRUE

Performance: Database Smart Flash Cache

Parameter change

- DB_FLASH_CACHE_FILE=file1, file1
- DB_FLASH_CACHE_SIZE=size1, size2

Performance: Temporary UNDO

View

- V\$TEMPUNDOSTAT

Parameter

- TEMP_UNDO_ENABLED=TRUE

Performance: Online Operations

Package

- DBMS_REDEFINITION.START_REDEF_TABLE (...,
copy_vpd_opt =>
DBMS_REDEFINITION.CONS_VPD_AUTO)
- EXEC DBMS_REDEFINITION.FINISH_REDEF_TABLE (...,
dml_lock_timeout => 100) ;

Miscellaneous: Partitioning

New Column

- DEF_INDEXING in
 - DBA_PART_TABLES
- INDEXING in
 - DBA_TAB_PARTITIONS
 - DBA_TAB_SUBPARTITIONS
 - DBA_INDEXES
- ORPHANED_ENTRIES in
 - DBA_INDEXES
 - DBA_IND_PARTITIONS

Package

- DBMS_PART.CLEANUP_GIDX

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font.

Miscellaneous: SQL

Parameter

- MAX_STRING_SIZE=standard|extended
- DB_SECUREFILE=preferred

Other PDB Creation Methods

ORACLE

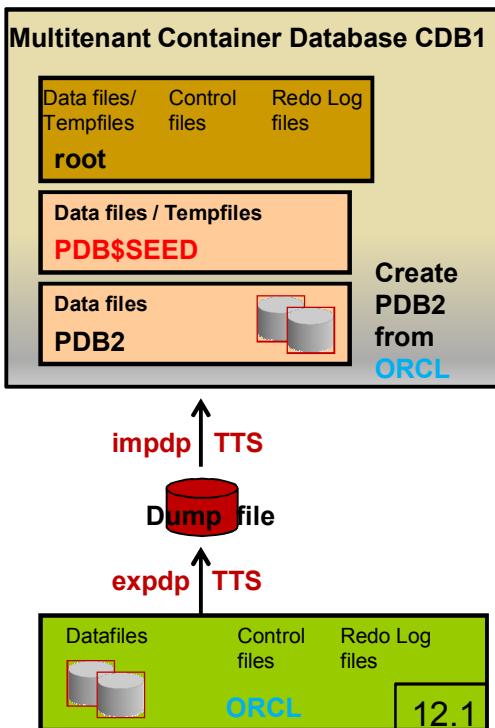
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This appendix explains how to create PDBs using methods that were not detailed in the lesson “*Creating Multitenant Container Databases and Pluggable Databases*”.

Note: For a complete understanding of the new multitenant architecture and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Administrator’s Guide 12c Release 1 (12.1)*
- *Oracle Database PL/SQL Packages and Types Reference 12c Release 1 (12.1) – “DBMS_PDB” chapter*

Plugging a Non-CDB into CDB Using Data Pump



1. Perform TTS or TDB export from **ORCL**
2. Connect to the root as a common user with **CREATE PLUGGABLE DATABASE** privilege.
3. Create new **PDB2** (Method 1).
4. Perform TTS or TDB import into **CDB1 / PDB2** with:
 - **ORCL dumpfile**
 - **ORCL datafiles**
6. Check application data:

```
SQL> CONNECT sys@PDB2
SQL> SELECT * FROM HR.EMP;
```

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If you choose the transportable tablespace (TTS) or database (TDB) export/import method, the method is the same as using Data Pump export/import to move data between two non-CDBs.

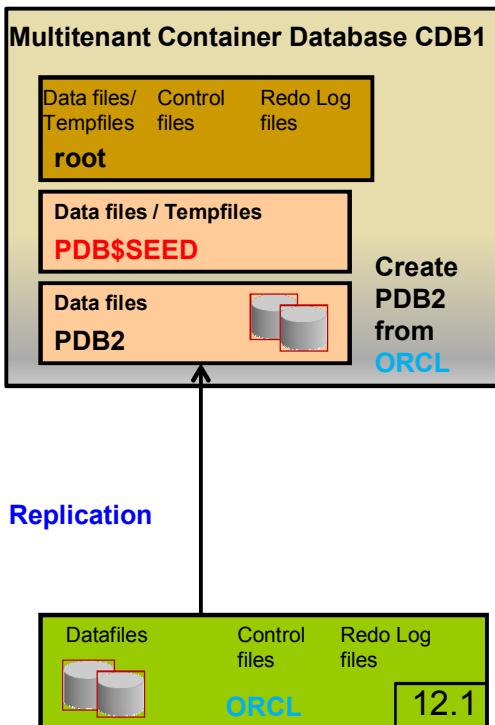
1. First perform a TTS or TDB export from **ORCL** database.
2. Then connect to the root of the CDB as a common user with **CREATE PLUGGABLE DATABASE** privilege
3. To create the new **PDB2** that will be the container for **ORCL** data. Use the method 1 as explained previously.
4. Finally, perform a TTS or TDB import into **PDB2** of CDB using both **ORCL** dump file and data files. The import command uses the connection to the **PDB2** service.
5. Then connect to **PDB2** to check with **dba_tables** view that your tables are in **PDB2**. Check that the application data has been imported.

```
SQL> SELECT * FROM HR.EMP;
```

If you cannot use transportable databases or tablespaces to move the data, then use a full conventional Data Pump export/import to move the data and metadata to the PDB.

Note: To get detailed information on how to perform a TDB, refer to the lesson covering Data Pump Enhancements or to the *Oracle Database Administrator's Guide 12c Release 1 (12.1) Chapter Transporting Data*.

Plugging a Non-CDB into CDB Using Replication



Method using replication:

1. Connect to the root as a common user with CREATE PLUGGABLE DATABASE privilege.
2. Create new PDB2 (Method 1).
3. Open PDB2 in read write mode.
4. Configure unidirectional replication environment from ORCL to PDB2.
5. Check application data:

```
SQL> CONNECT sys@PDB2
SQL> SELECT *
  2 FROM dba_tables;
SQL> SELECT * FROM HR.EMP;
```

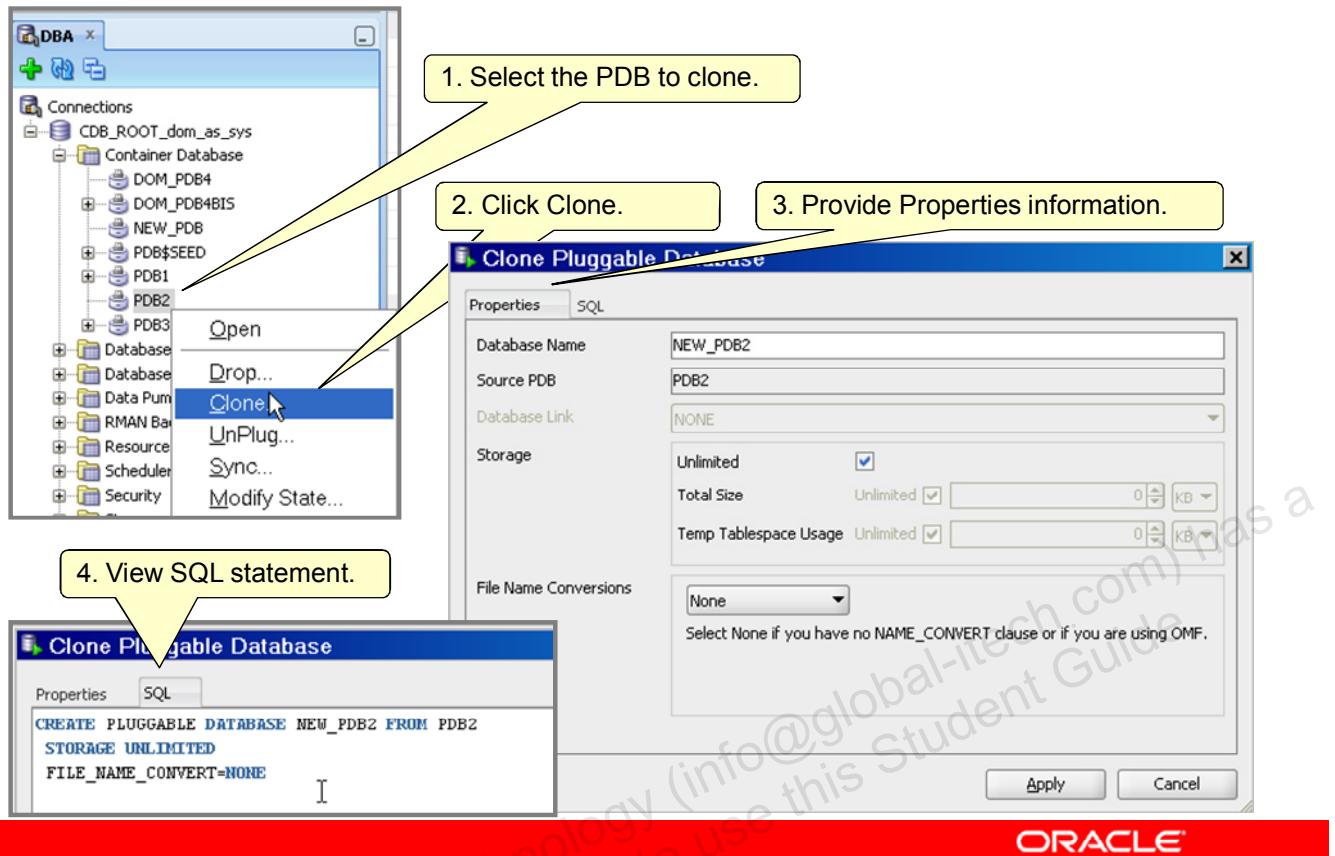
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

If you choose the replication method, the steps would be the following:

1. Connect to the root as a common user with CREATE PLUGGABLE DATABASE privilege
2. Use method 1 as explained previously to create the new PDB2 that will be the container for ORCL data.
3. Open PDB2 in read-write mode.
4. Configure an Oracle GoldenGate unidirectional replication environment with the non-CDB ORCL as the source database and the PDB2 as the destination database.
5. When the data at PDB2 catches up with the data at the non-CDB ORCL, fail over to PDB2.

Cloning PDBs: Using SQL Developer



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

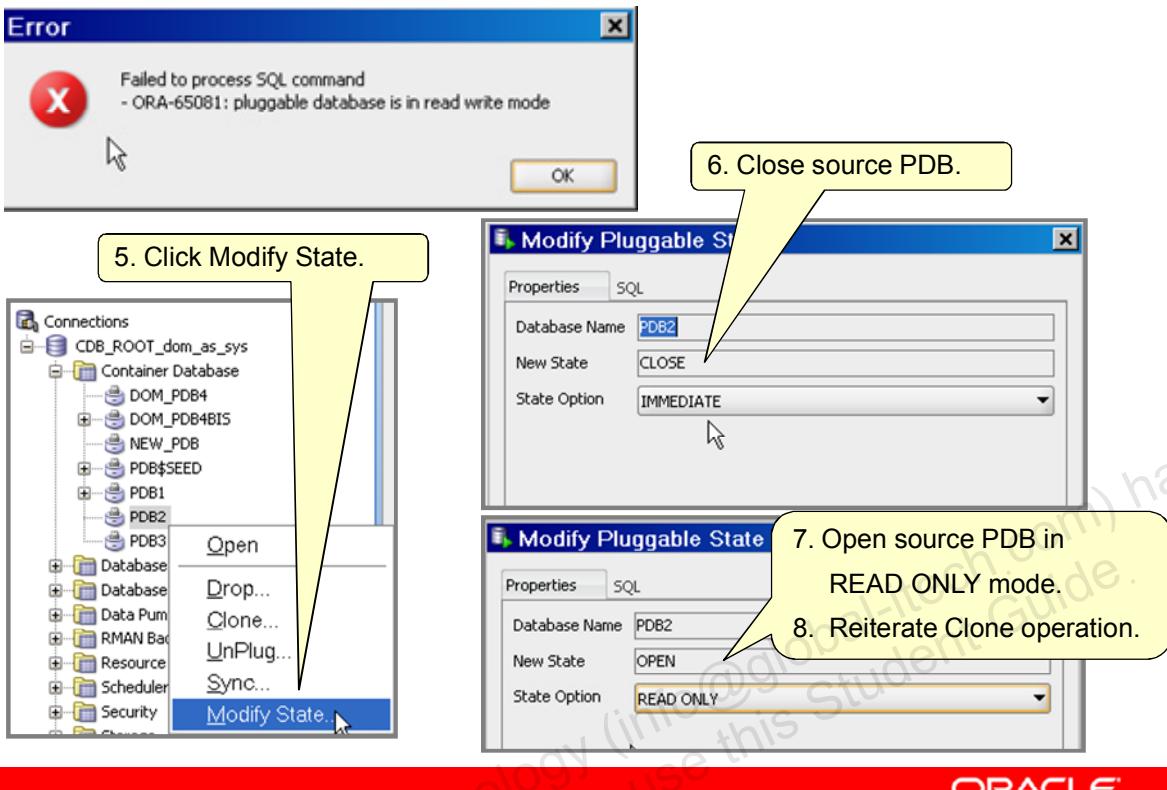
ORACLE

You can use SQL Developer to clone a PDB from another PDB.

When you clone a PDB from another PDB, you first select the PDB to clone, then choose the Clone option, then provide the new PDB name, the PDB administrator username and the target files location in the File Name Conversions attribute. Use Custom Names to provide the Target File location. The Source File displays the list of the datafiles used as templates during the copy.

Cloning PDBs: Using SQL Developer

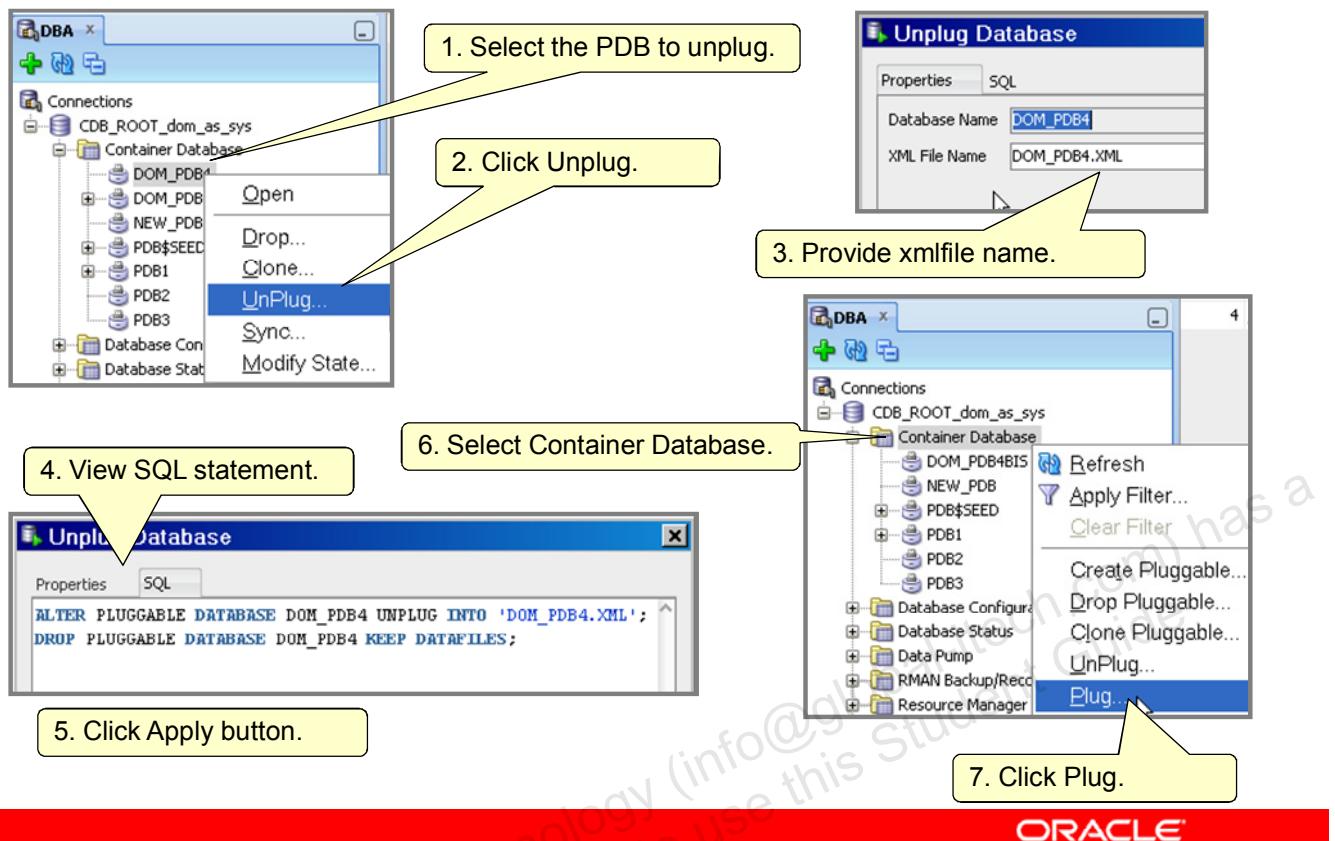
The source PDB must be in READ ONLY mode.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

Plugging Unplugged PDB: Using SQL Developer



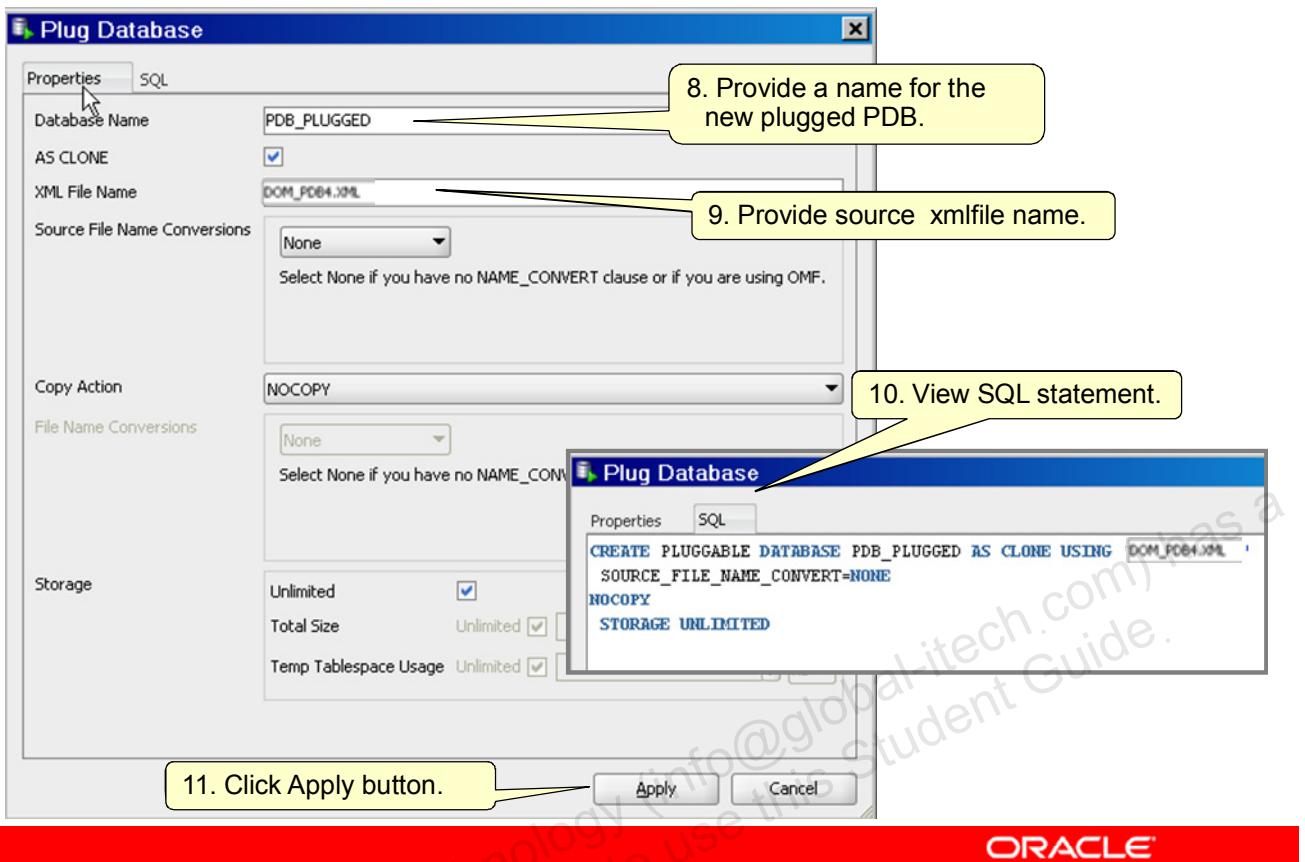
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

You can use SQL Developer to perform the unplug and plug operations.

First select the PDB to unplug. Then click Unplug option. Provide a name for the xmlfile file that will be generated. You can view the SQL statement before applying it. Then to plug the unplugged PDB, select the Container Database and choose Plug option.

Plugging Unplugged PDB: Using SQL Developer


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To plug the PDB, provide a name for the new plugged PDB and the XML file source file. Before applying the statement, you can view it.

The last action would be the PDB opening.

