

Oracle Database 12c: New Features for Administrators

Student Guide - Volume I

D77758GC10

Edition 1.0

May 2013

D80604

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

I

Introduction



ORACLE

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

Overview

- This course focuses on the features of Oracle Database 12c that are applicable to database administration.
- Previous experience with Oracle databases (particularly Oracle Database 11g) is required for a full understanding of many of the new features.
- Hands-on practices emphasize functionality rather than test knowledge.

The Oracle logo, consisting of the word "ORACLE" in a white sans-serif font inside a red horizontal bar.

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

This course is designed to introduce you to the new features of Oracle Database 12c that are applicable to the work usually performed by database administrators and related personnel. The course does not attempt to provide every detail about a feature or cover aspects of a feature that were available in previous releases (except when defining the context for a new feature or comparing past behavior with current behavior). Consequently, the course is most useful to you if you have administered other versions of Oracle databases, particularly Oracle Database 11g. Even with this background, you should not expect to be able to implement all of the features discussed in the course without supplemental reading, especially the Oracle Database 12c documentation.

The course consists of instructor-led lessons and demonstrations, plus many hands-on practices that enable you to see for yourself how certain new features behave. As with the course content in general, these practices are designed to introduce you to the fundamental aspects of a feature. They are not intended to test your knowledge of unfamiliar syntax or to provide an opportunity for you to examine every nuance of a new feature. The length of this course precludes such activity. Consequently, you are strongly encouraged to use the provided scripts to complete the practices rather than struggle with unfamiliar syntax.

Oracle Database Innovation

... continuing with

Oracle Database 12c

... with Oracle Database 11g

... with Oracle
Database 10g

Audit Vault

Database Vault

Secure Enterprise Search

Grid Computing

Automatic Storage Mgmt

Self Managing Database

XML Database, Oracle Data Guard, RAC, Flashback Query, Virtual Private Database

Built-in Java VM , Partitioning Support, Built-in Messaging, Object Relational Support, Multimedia Support

Private DB Cloud

Defense in Depth

Information Lifecycle Mgt

Extreme Availability

Flex Clusters

Performance and Ease of Use

Oracle Grid Infrastructure

Real Application Testing

Automatic SQL Tuning

Fault Management

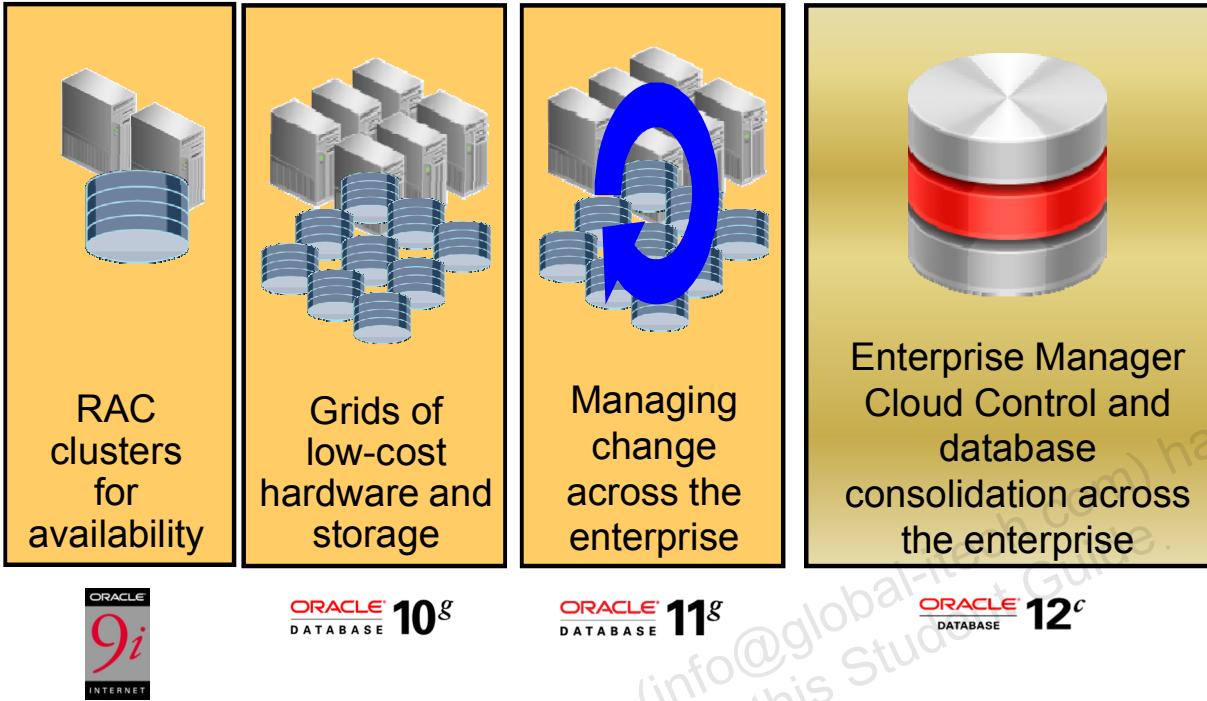
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

As a result of its early focus on innovation, Oracle has maintained the lead in the industry with a large number of trend-setting products.

Some of the marquee areas in the Oracle Database 12c release are the following:

- Private Database Cloud
- Defense in Depth included Oracle Data Redaction, Real Application Security
- Information Lifecycle Management (ILM), which includes hot/cold data classification, declarative compression and tiering, In-database Archiving and Valid-Time Temporal
- Flex Clusters
- Extreme Availability, which includes Data Guard Far-Sync and Application Continuity
- Lower Cost Migrations
- Performance and Ease of Use, which includes “just in time” optimizations, attribute clustering, and zone maps for Exadata only

Enterprise Cloud Computing



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



Oracle Database 10g was the first database management system designed for grid computing.

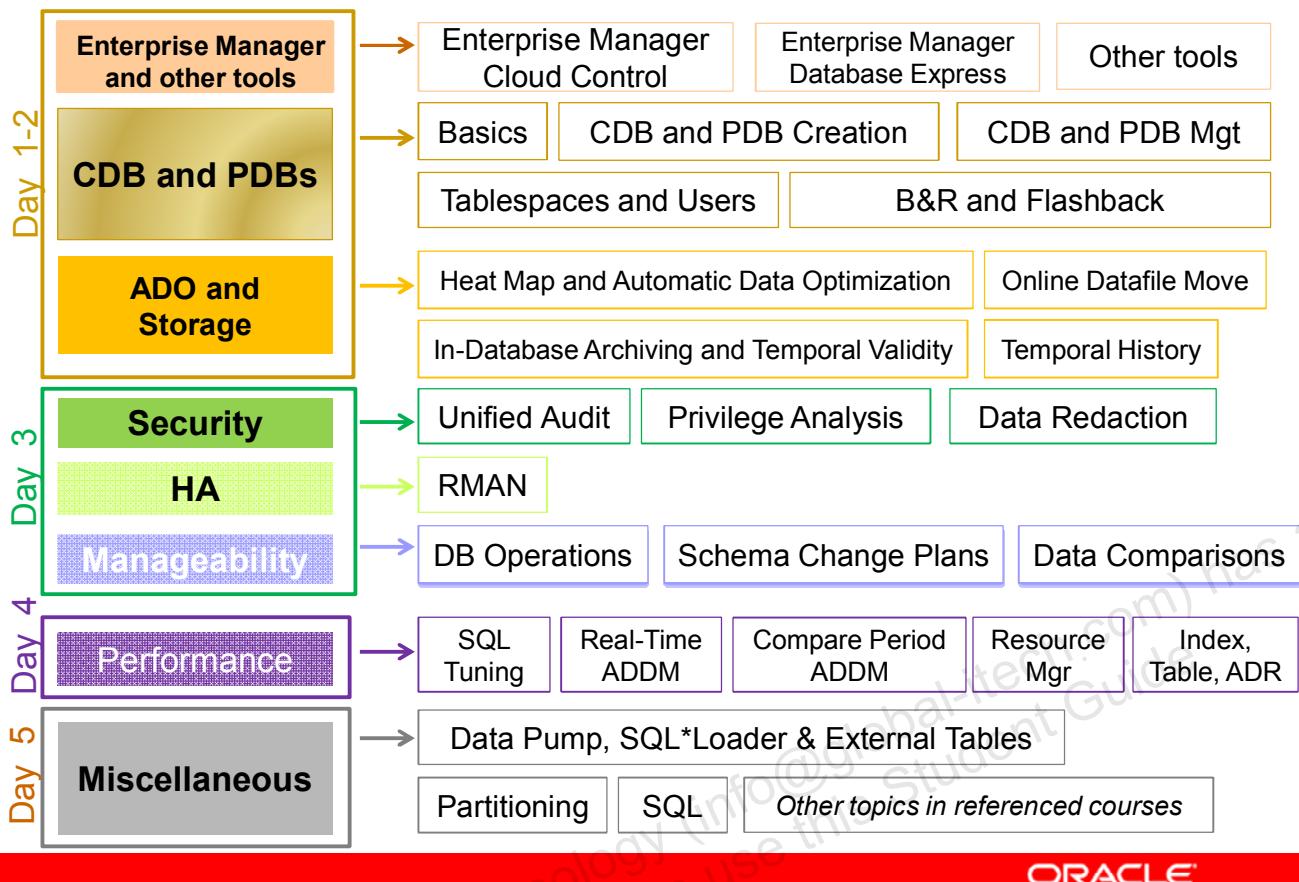
Oracle Database 11g consolidates and extends Oracle's unique ability to deliver the benefits of grid computing, transforming data centers from silos of isolated system resources to shared pools of servers and storage.

Oracle Database 12c and Enterprise Manager Cloud Control are designed for cloud computing. Cloud computing creates a complete, pre-integrated, off-the-shelf private cloud solution that allows you to quickly transform the enterprise data center into a private cloud.

The key benefits are the following:

- Reduce servers sprawl and improve CPU utilization by consolidating on fewer servers
- Reduce the amount of time a DBA spends installing and configuring databases, by automating deployment of standard database configurations
- Single console to manage entire Cloud life cycle – plan, set up, deliver, operate
- Prevent resource hogging by setting quotas for individual users
- Forecast future resource needs by analyzing trending reports
- Compute chargeback based on performance and configuration metrics

Oracle Database 12c New and Enhanced Features



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

Features, Options and Schedule

Enterprise Manager Cloud Control: Oracle Enterprise Manager is Oracle's integrated enterprise IT management product line, and provides the industry's first complete cloud lifecycle management solution.

Oracle Multitenant: Oracle Multitenant is a new option in Oracle Database 12c. The multitenant architecture enables an Oracle database to contain a portable set of schemas, objects, and related structures that appears logically to an application as a separate database. This self-contained collection is called a pluggable database (PDB). A multitenant container database (CDB) contains one or several PDBs.

Information Lifecycle Management: One challenge facing each organization is to understand how its data evolves and grows, monitor how its usage changes over time, and decide how long it should survive, while adhering to all the rules and regulations that now apply to that data. Information Lifecycle Management (ILM) is designed to address these issues, with a combination of processes, policies, software, and hardware so that the appropriate technology can be used for each stage in the life cycle of the data. ILM offers new features including heat Map, Automatic Data Optimization and enhancements to In-Database Archiving, including the new Temporal Validity feature.

Defense in Depth

- Unified auditing makes audit information available in a uniform format and centralized in a single place accessible only by privileged users.
- New administrative privileges help separation of duties.
- Privilege Analysis contributes to identify privilege usage and to decide which privileges must be revoked.
- Oracle Data Redaction prevents the display of sensitive data by performing masking in each application, defining redaction at the database level and displaying redacted data regardless of the access method, even to DBAs.

High Availability

RMAN enhancements include recovering a table or table partitions from existing backups, using backup sets for cross-platform data transport, connecting with the new `SYSBACKUP` privilege, and many more.

Manageability

- Database Operation monitoring extends and generalizes Real Time SQL Monitoring.
- Database Change 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 done in a production environment that are causing problems. Enterprise Manager with Schema Change Plans and Data Comparisons 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.

Performance

- SQL tuning is essentially based on performance automation with Adaptive SQL Plan Management, Adaptive Execution Plans and Optimizer Statistics Management.
- Real-Time ADDM and Emergency Monitoring detect that the system is sick and analyzes the root causes of the hanging situation to help you find the appropriate action.
- In a situation where the performance has decreased or increased, Compare Period Advisor identifies what has changed and why and which changes may have impacted the performance between two periods of time. This helps you take appropriate actions upon relevant analysis.
- Resource Manager is now adapted to multitenant container databases and pluggable databases to share CPU and other resources among the different containers in a database.

There are also many other enhancements about Oracle Data Pump, SQL*Loader, partitioning and online operations, and SQL covered in the course.

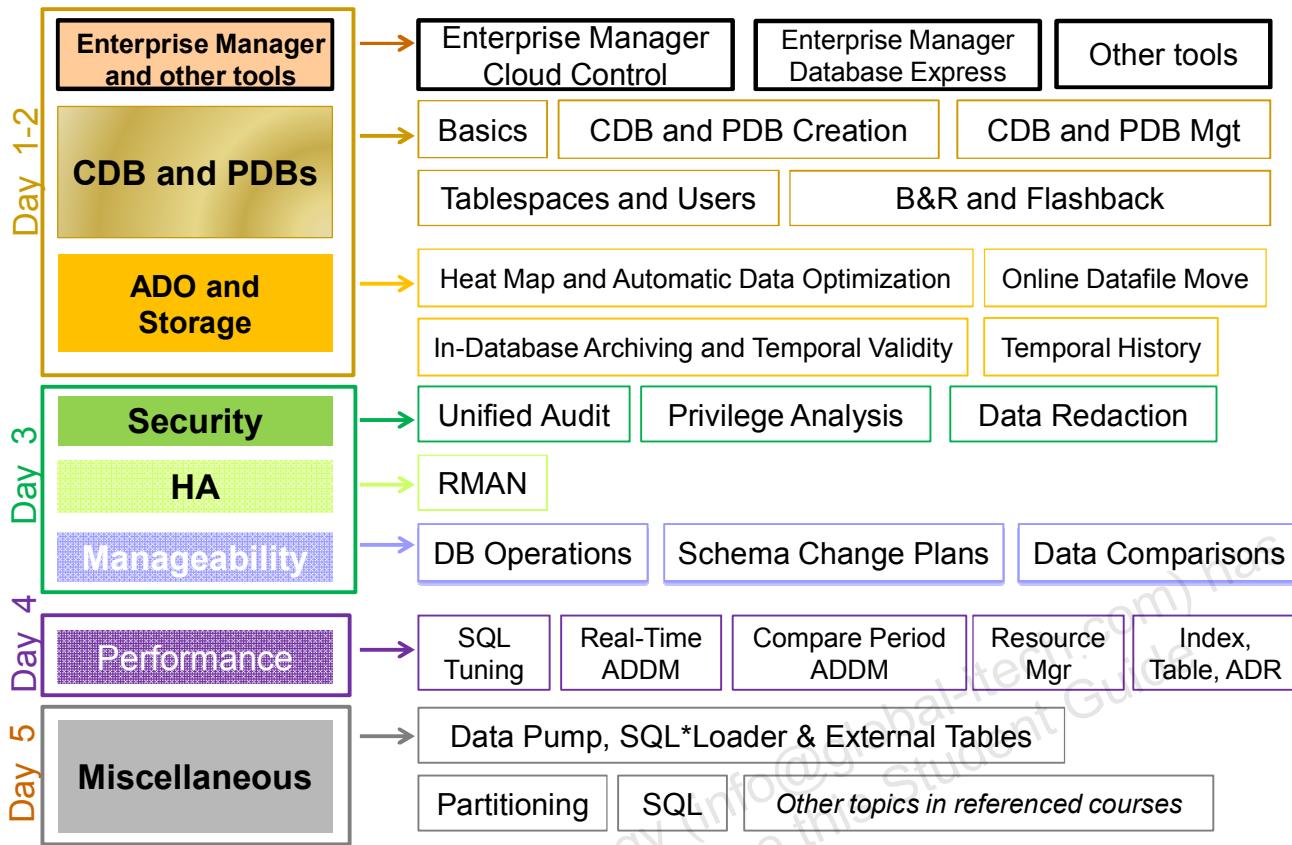
Many other subjects like Automatic Storage Management, Real Application Cluster, Data Guard, Real Application Testing and many other subjects of expertise cannot be presented in this course. You can find references to these courses, demonstrations, and OBEs in the last lesson.

1 **Enterprise Manager Cloud Control and Other Tools**

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.

Objectives

After completing this lesson, you should be able to:

- Describe the different components of Cloud Control
- Explain the architecture of Cloud Control
- List the target types managed by Cloud Control
- Explore the Oracle Enterprise Manager Cloud Control interface
- Describe Oracle Enterprise Manager Database Express navigation
- Use DBCA to create new types of databases
- Explore the Oracle SQL Developer interface



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

Note

For a complete understanding of Oracle Enterprise Manager Cloud Control and Database Express installation and usage, refer to the following guides in the Oracle documentation:

- *Oracle Enterprise Manager Cloud Control Basic Installation Guide 12c Release 1*
- *Oracle Enterprise Manager Cloud Control Advanced Installation and Configuration Guide 12c Release 1*
- *Oracle Enterprise Manager Cloud Control Administrator's Guide 12c Release 1*
- *Oracle Enterprise Manager Licensing Information 12c Release 1*

Key Challenges for Administrators

As the composition of the data center broadens into the cloud environment, the challenges to manage it also increase. The key challenges for managing a data center include:

- Monitoring performance and availability
- Resolving problems quickly
- Containing operating costs
- Aligning IT with business priorities

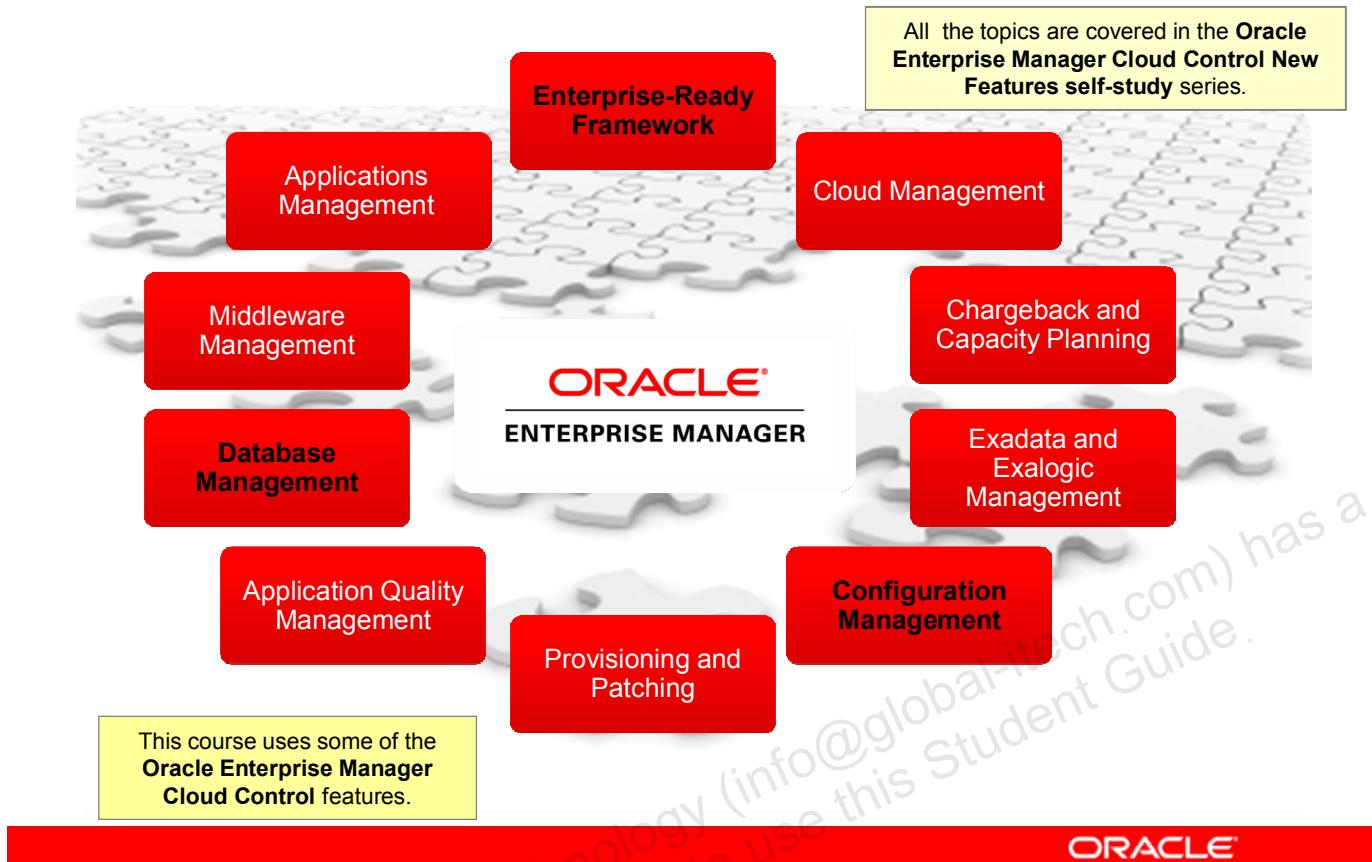


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

As the data center grows with the growth in business, so do the challenges. An administrator is faced with challenges that include:

- Monitoring high levels of performance and availability of applications
- Identifying and resolving problems quickly and effectively
- Enabling IT professionals to use resources effectively, thereby reducing costs
- Aligning IT with business priorities to ensure that businesses are agile enough to meet the changing needs

Enterprise Manager Cloud Control



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

ORACLE

Key Objectives in Enterprise Manager Cloud Control Design

- Designing a management framework that is capable of providing next-generation functionality
- Enhancing application-to-disk manageability
- Providing a complete enterprise private cloud solution

Enterprise Manager Cloud Control includes the following features:

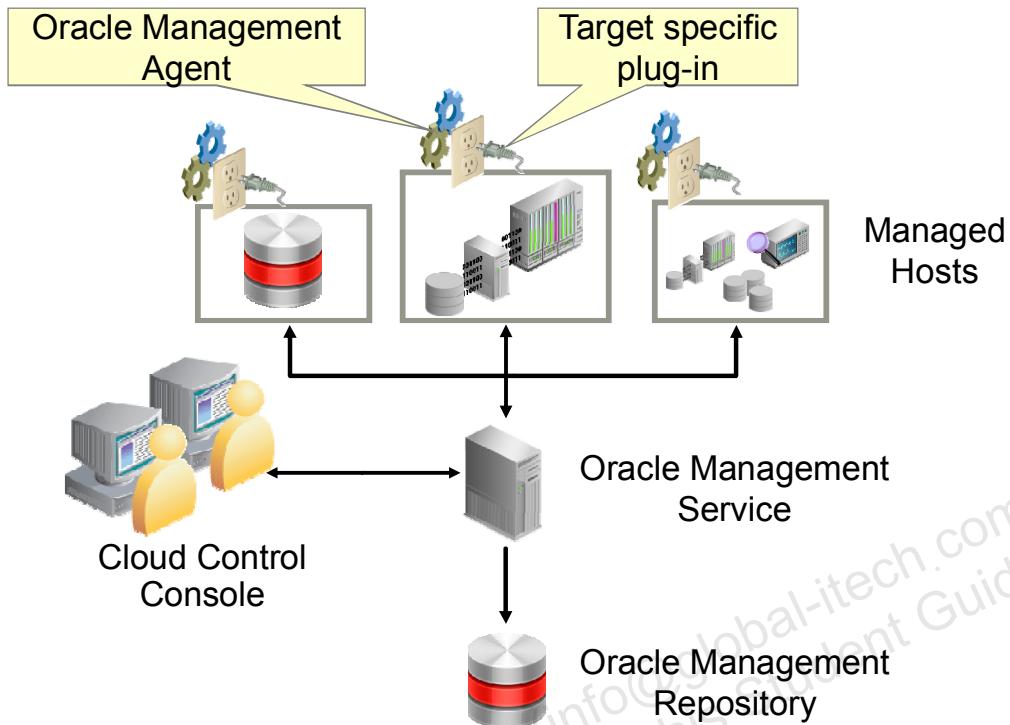
- **Enterprise-Ready Framework:** Provides modular and extensible architecture, target plug-ins, self-updateable entities, integrated Support Workbench, and centralized incident console
- **Cloud Management:** Provides complete cloud lifecycle management
- **Chargeback and Capacity Planning:** Provides chargeback based on target types, and uses Automatic Workload Repository (AWR) Warehouse to consolidate AWR reports from multiple databases across the enterprise
- **Exadata and Exalogic Management:** Provides an integrated view of the hardware and software in an Exadata machine, and complete lifecycle management for Exalogic systems
- **Configuration Management:** Provides an integrated set of tools, agent-less discovery, integration with My Oracle Support, and custom configuration capabilities

- **Provisioning and Patching:** Provides profiles for provisioning known configurations, user-defined deployment procedures, and a software library integrated with self-updating capabilities
- **Application and Quality Management:** Database Replay, Application Server Replay, Real Application Testing integrated with Data Masking, and test database management including Application Data Model
- **Database Management:** Provides management of Oracle Database systems, including performance management and change lifecycle management. Some aspects of Database Management are covered in detail in this course.
- **Middleware Management:** Provides management of Fusion Middleware systems
- **Applications Management:** Provides management of Fusion Applications

Note: For a complete understanding of Oracle Enterprise Manager Cloud Control usage, refer to the following sources of information:

- *Using Oracle Enterprise Manager Cloud Control* course
- *Oracle Enterprise Manager Cloud Control: Install and Upgrade* course
- *Oracle Enterprise Manager Cloud Control New Features* Self-Study series
- *Oracle Enterprise Manager 12c* demonstrations under Oracle Learning Library (URL www.oracle.com/goto/oll)

Cloud Control Components



ORACLE

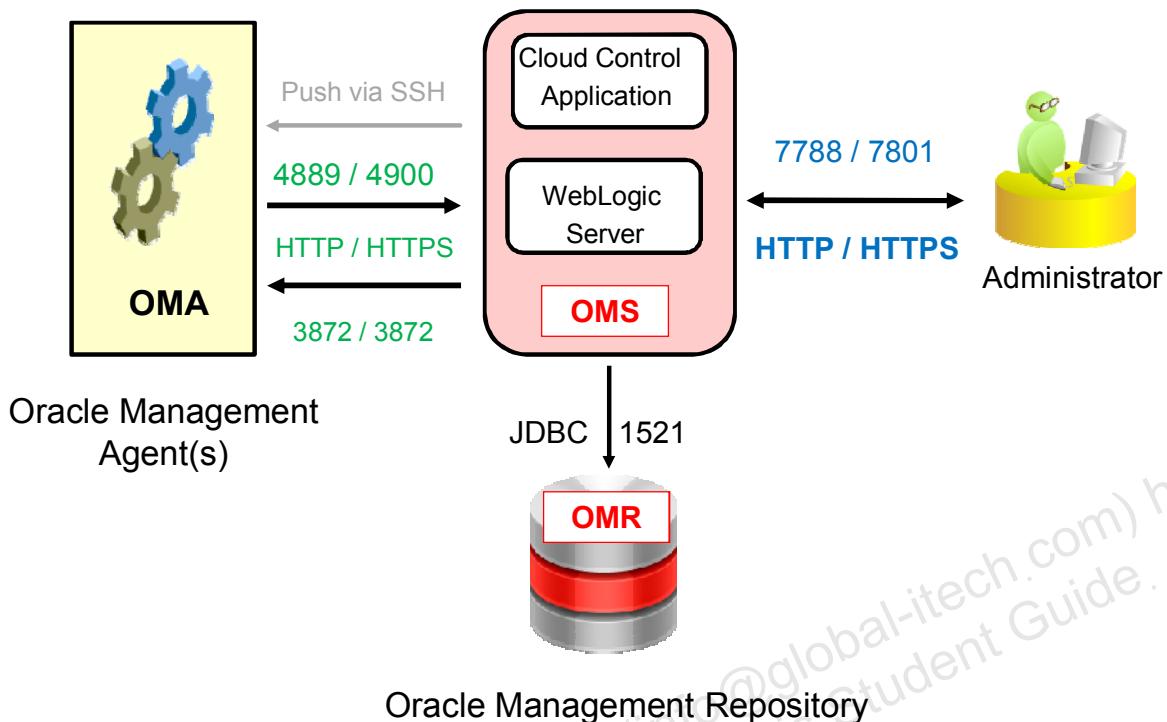
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Enterprise Manager Cloud Control is composed of four main components as illustrated:

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

The Oracle Management Agent runs on hosts, gathering metric data about those host environments as well as using plug-ins to monitor availability, configuration, and performance and to manage targets running on the host. The agents communicate with the Oracle Management Service to upload metric data collected by them and their plug-ins. In turn, the OMS stores the data it collects in the Oracle Management Repository where it can be accessed by the OMS for automated and manual reporting and monitoring. The OMS also communicates with the agents to orchestrate the management of their monitored targets. As well as coordinating the agents, the OMS runs the Cloud Control Console web pages that are used by administrators and users to report on, monitor, and manage the computing environment that is visible to Cloud Control via the agents and their plug-ins.

Components and Communication Flow



ORACLE

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

The communication flow between the Cloud Control components is illustrated using directional arrows. Communication between the Agent and the OMS, and the OMS and the console is bi-directional. All the ports shown and listed in the slide are default values that can be changed during installation, either by the installer as it searches for available ports, or explicitly by you. You can also change ports after installation.

- The Agent uploads data to the OMS via HTTP on port 4889 or via HTTPS on port 4900. (Designed to be able to work with WAN.)
- The OMS communicates with the Agent via HTTP or HTTPS on port 3872.
- The reason for the separate ports for OMS to OMA communications is that they can communicate asynchronously and simultaneously to one another.
- The OMS communicates with the OMR via JDBC on port 1521. Although the OMR will return data to the OMS, this is not considered to be a separate communication between the two; hence, the flow is shown to be unidirectional from OMS to OMR.
- Cloud Control console users access the Cloud Control webpages via HTTPS on port 7801 or via HTTP on port 7788.

Knowing the ports used in your Cloud Control installation is important, especially if you are managing hosts behind firewalls or where other network restrictions apply, because communication will need to be allowed on these ports and in the directions shown.

Oracle Management Repository

The Oracle Management Repository (OMR):

- Resides in an Oracle database
- Includes schema objects belonging to SYSMAN
- Must be installed in a pre-existing database
- Can be installed in a RAC database

Note: Uses a restricted-use license of the Oracle Database



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

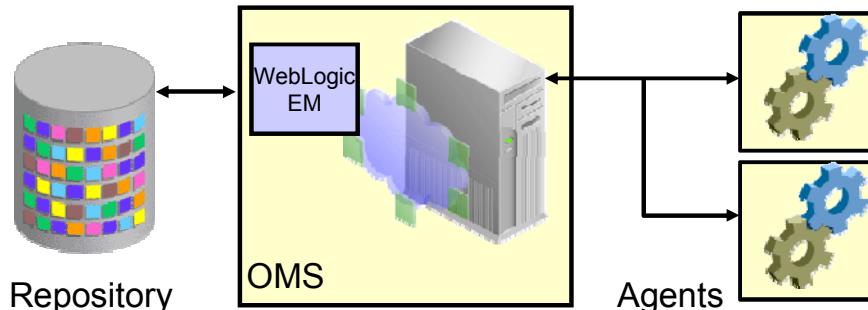
The OMR is installed in an Oracle database as a group of approximately 4,000 schema objects belonging to the SYSMAN user stored in three tablespaces: MGMT_ECM_DEPOT_TS, MGMT_TABLESPACE, and MGMT_AD4J_TS. These schema objects contain information about Enterprise Manager Cloud Control users and administrators, targets and applications that are monitored and managed by Enterprise Manager Cloud Control, and groups, systems, incidents, and other Enterprise Manager Cloud Control artifacts. The OMR is created during installation in a preexisting database, and for scalability requirements can be installed in a Real Application Clusters (RAC) database. In this case, customers must license the second node for the database, and both nodes require an Oracle Real Application Clusters license.

The database used to house the OMR should not be used for any other applications for the following reasons:

- Enterprise Manager Cloud Control's usage of the database should not have to compete with any other usage.
- Using the OMR database for other applications may restrict your ability to upgrade and patch the OMR schema and database as required
- Enterprise Manager Cloud Control includes a restricted-use database license that can be used for the OMR only

Note: Refer *Oracle Enterprise Manager Licensing Information 12c Release 1*, Section *Enterprise Manager Restricted-Use License*

Controlling the Enterprise Manager Cloud Control Framework



Component Control Utilities		
Repository	OMS	Agent
sqlplus or srvctl	emctl	emctl
lsnrctl		

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

Each component of the Enterprise Manager Cloud Control framework has its own utility or utilities that can be used to monitor, start, and stop the component. In many cases, these utilities also provide some capability to configure the component beyond the simple start-and-stop functionality.

RAC databases require the use of the Server Control commands; for single instances, there is a choice between SQL*Plus and Server Control. Server Control is usable when Oracle Restart is installed and the database is registered with the OLR.

To start and stop the listener, either use the Server Control utility or the lsnrctl command.

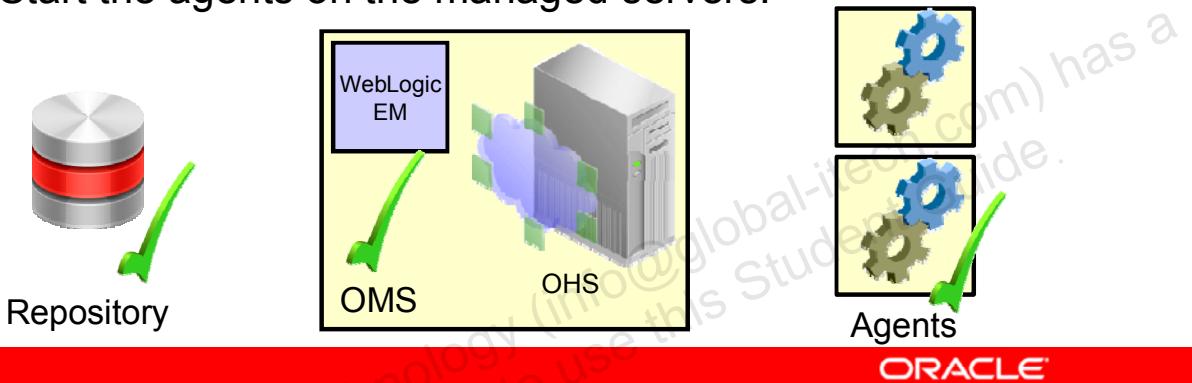
Examples:

```
srvctl stop database -d orcl -o immediate
srvctl start database -d orcl -o open
```

Starting the Enterprise Manager Cloud Control Framework

To start the Cloud Control framework, perform the following steps:

1. Start the repository database listener.
2. Start the repository database instance.
3. Start the OMS.
4. Start the agent on the OMS/repository server.
5. Start the agents on the managed servers.



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

To start the whole Enterprise Manager Cloud Control framework, follow the steps below:

1. Start the repository listener:

```
$ORACLE_HOME/bin/lsnrctl start
```
2. Start the repository database instance:

```
$ORACLE_HOME/bin/sqlplus / as sysdba  
    SQL> startup
```
3. Start the OMS (including OHS and WebLogic Managed Server):

```
$OMS_HOME/bin/emctl start oms
```
4. Start the agent (on OMS/repository host):

```
$AGENT_HOME/bin/emctl start agent
```
5. Start the agent on the managed servers:

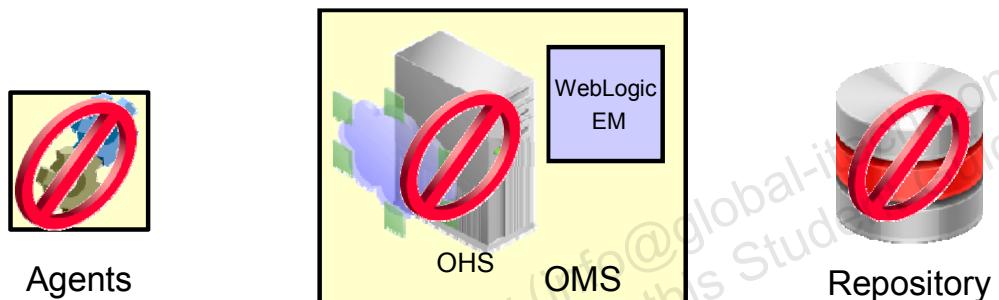
```
$AGENT_HOME/bin/emctl start agent
```

Note: Use the `SRVCTL` command if you have a RAC instance for the repository.

Stopping the Enterprise Manager Cloud Control Framework

To stop the Enterprise Manager Cloud Control framework, perform the following steps:

1. Stop the agents on managed servers.
2. Stop the agent on the OMS/repository server.
3. Stop the OMS.
4. Stop the repository database instance.



ORACLE

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

To stop the whole Enterprise Manager Cloud Control framework, follow the steps below:

1. Stop the agent on the managed servers:

```
$AGENT_HOME/bin/emctl stop agent
```
2. Stop the agent (on OMS/repository host):

```
$AGENT_HOME/bin/emctl stop agent
```
3. Stop the OMS (including OHS and WebLogic Managed Server):

```
$OMS_HOME/bin/emctl stop oms
```
4. Stop the repository database instance:

```
$ORACLE_HOME/bin/sqlplus / as sysdba  
SQL> shutdown immediate
```

Note: Use the SRVCTL command if you have a RAC instance for the repository.

Different Target Types

Enterprise Manager Cloud Control can monitor, administer, maintain, and manage many different types of targets including:

- Oracle Databases
- Oracle Database Listener
- Oracle Fusion Middleware products
- Oracle Application Server
- Oracle WebLogic Server
- Oracle applications, including E-Business Suite, SOA, Siebel, and PeopleSoft
- Exadata and Exalogic
- Cloud Control Components: OMR and OMS
- Third-party products

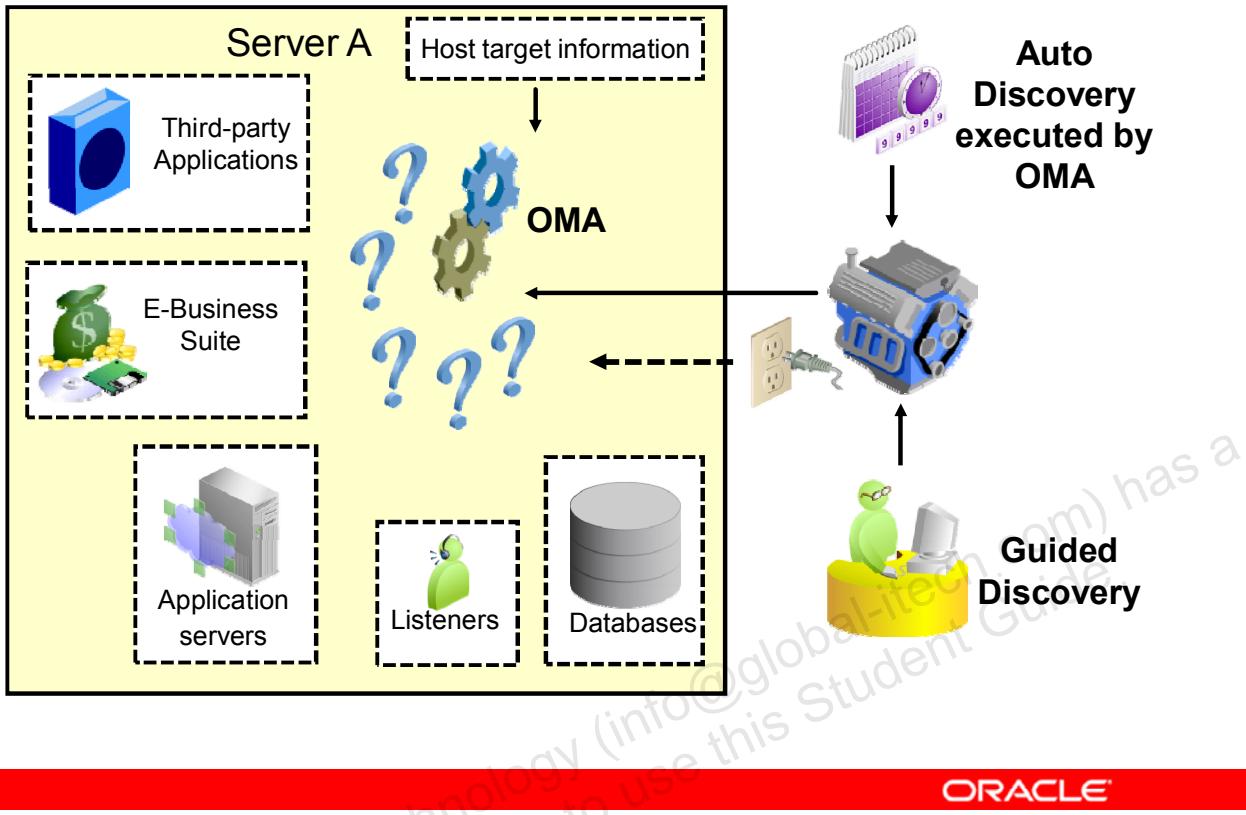


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Targets are the entities that Enterprise Manager Cloud Control manages. To do so, it uses target-type specific plug-ins and host-specific agents.

Enterprise Manager Cloud Control can monitor, administer, maintain, and manage different types of targets as listed in the slide. As your environment changes, you can add and remove targets from Enterprise Manager Cloud Control as needed. The commonly used Oracle targets (including Enterprise Manager Cloud Control components, such as the OMR and OMS) are predefined as part of the base Enterprise Manager Cloud Control product, but Enterprise Manager Cloud Control has an open API that enables you to create custom targets.

Target Discovery



ORACLE

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

After the Agent has been installed on a host, it needs to look for targets that it can manage. As an Enterprise Manager Cloud Control administrator, you can guide that process from the Enterprise Manager Cloud Control console pages. Guided discovery allows you to nominate a family of target types that you want to search for, such as database and listeners, and then the agents where you want that search to be executed. If any new targets are discovered, the appropriate plug-in will be pushed from the OMS if it is not already installed on the agent, the target will be recorded in the OMR, and monitoring will begin.

You can also configure auto discovery to run at regular intervals and get an agent to search for known targets unattended, allowing you to review the results at a later stage and promote discovered targets to become managed targets.

Enterprise Manager Cloud Control

ORACLE

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

The image in the slide is of the Enterprise Summary page of Oracle Enterprise Manager Cloud Control. The user interface (UI) functionality includes:

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

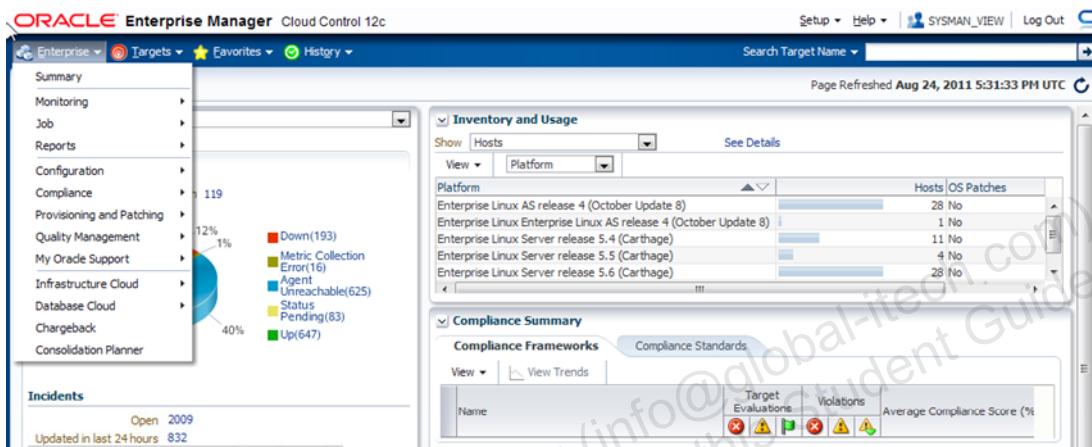
User Interface

Setting your home page:

- Predefined home page based on roles
- Any page

Menu-based navigation:

- Make any page a favorite for quick access.



ORACLE

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

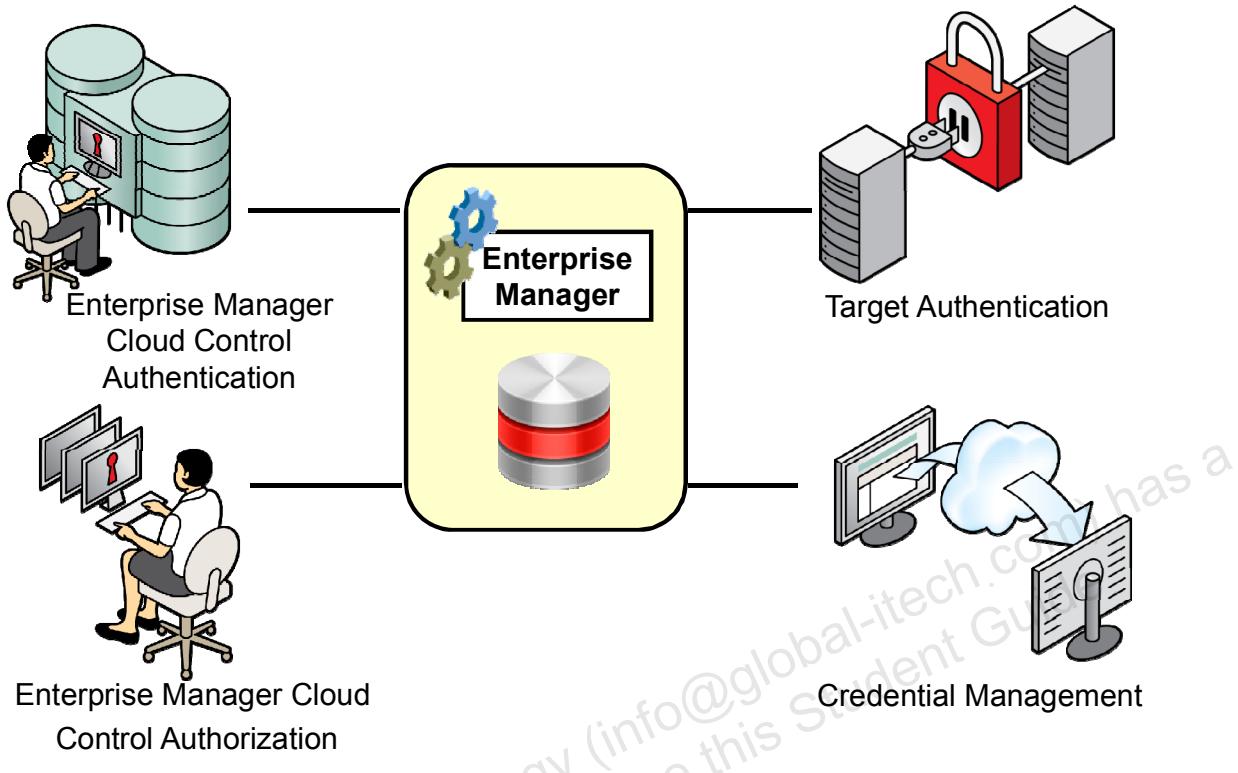
User Interface Enhancements

The user interface in Enterprise Manager Cloud Control has been rewritten in Application Development Framework (ADF). When you log in to the new UI, use drop-down menus to navigate from one place to another in the product.

- Choose your own home page: When you first log in to Enterprise Manager, you are provided with a selection of pre-defined home pages based on roles. If you are managing databases, you can choose the database home page. If those are not suitable, you can select any page in the UI to be your home page instead.
- Mark any page as a “favorite” for quick access. Because you manage certain targets frequently, you mark these target home pages as favorites in much the same way you mark a favorite in a browser. However, because the favorites you mark in Enterprise Manager are stored in the repository, you can move from client machine to client machine and your favorites are still available to you.

For information about how to customize your Enterprise Manager Cloud Control console, follow the demonstration « *Oracle Enterprise Manager 12c: Console Overview and Customization* under Oracle Learning Library.

Security: Overview



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

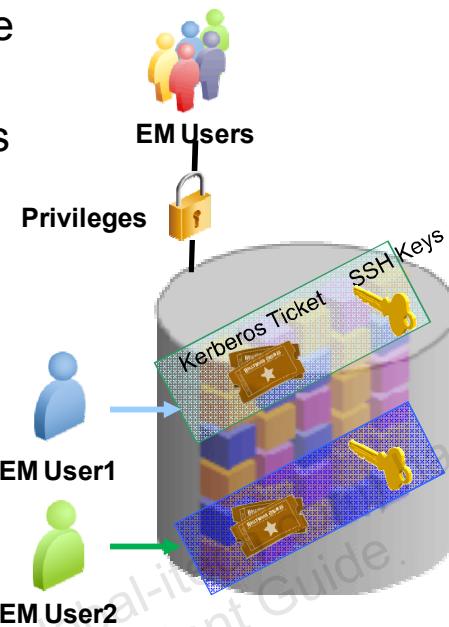
Enterprise Ready Framework: Security

The Enterprise Manager security system can be divided into four parts (as shown in the graphic):

- Enterprise Manager Cloud Control authentication
- Enterprise Manager Cloud Control authorization and privileges
- Credential management (host credentials can be defined here)
- Target authentication (host credentials can be used here)

Managing Securely with Credentials

- Centralized credential store for ease of management
- Support for managing passwordless and strong authentication credentials
 - Kerberos tickets
 - SSH keys
- Reuse and sharing among users (without disclosing the sensitive content of credentials)
- Controlled and protected access
- Support for sudo/powerbroker



ORACLE

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

Security with Credentials

As a management tool that handles a lot of scripts and powerful actions like patching, Enterprise Manager has to work with a lot of credentials for hosts, databases, and a range of other objects. Managing all these credentials can be a real challenge.

- The centralized store facilitates this task because you can name and store credentials there.
- Passwordless and strong authentication credentials are supported, such as the Kerberos tickets and SSH key pairs.
- Credentials can be reused and shared among users (without disclosing the sensitive content like a password). Users are granted access to these credentials by the use of privileges, and so they can be reused without knowing what the contents of the credentials themselves are.
- Access to the credentials is controlled and protected by privileges.
- The Enterprise Manager credential subsystem enables you to securely store credentials as preferences or operation credentials, which can then be used to perform different system management activities. Enterprise Manager also supports sudo/powerbroker-based impersonation.

Distinguishing Credentials

- Named credentials
- Preferred credentials
- Default credentials
- Access level:
 - **View:** access to use the credential
 - **Edit:** to change the credential (including changing its name and password)
 - **Full:** for complete access (including the ability to delete the credential)
- Usage classification: Job, collection, and monitoring



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

Types of Credentials

As the Enterprise Manager administrator, you can also store credentials (username/password, a public key-private key pair, or an X509v3 certificate) as **named credentials** in Enterprise Manager to use when performing operations like running jobs, patching, and other system management tasks. Objects refer or point to named credentials. They are “placeholders” to facilitate, for example, the changing of passwords.

You can store, access, and modify a fixed number of username/password-based credentials as **preferred credentials** to simplify access to managed targets by storing target login credentials in the Management Repository.

Default credentials can be set for a particular target type and will be available for all the targets of the target type.

The three levels of access that can be granted are:

- **View access:** To use the credentials
- **Edit access:** To change the credentials, including changing its name and password
- **Full access:** For complete access, including the ability to delete the named credential

Credentials can also be classified by their usage, such as job credentials (used by the job system), collection credentials, and monitoring credentials (used by OMA).

Core concepts and definitions:

- **Credential type** is the type of authentication supported by a target type. For example, a host can support a username/password-based authentication, public key authentication, or Kerberos authentication. Various authentication schemes are supported, including native agent authentication and SSH.
- **Credential set** is a placeholder for a credential. Credential sets can be used to decouple credentials from the system that uses a credential. A credential set enables you to change its mapping to named credentials for a target without editing the system that uses the credential. For example, you could have a credential set for patching tasks.
- **Credential store** is a logical store for all the named credentials of an Enterprise Manager administrator.

Defining credentials by:

- **Credential name:** The credential is referenced using the name of the credential in the credential store.
- **Credential set:** The credential is referenced using the credential set name and the target name. The lookup gets the credential associated with the credential set name and target name.
- **Direct value:** The credential is specified by providing the values of the attributes. This reference does not refer to a credential in the credential store.

For information about how to set credentials, follow the demonstrations:

- *Oracle Enterprise Manager 12c: Create and Use Named Credentials* under Oracle Learning Library
- *Oracle Enterprise Manager 12c: Create SSH Key Named Credentials* under Oracle Learning Library
- *OBE Enterprise Ready Framework: Create and Use Credentials*

Quiz

Which targets can be managed by using Enterprise Manager Cloud Control?

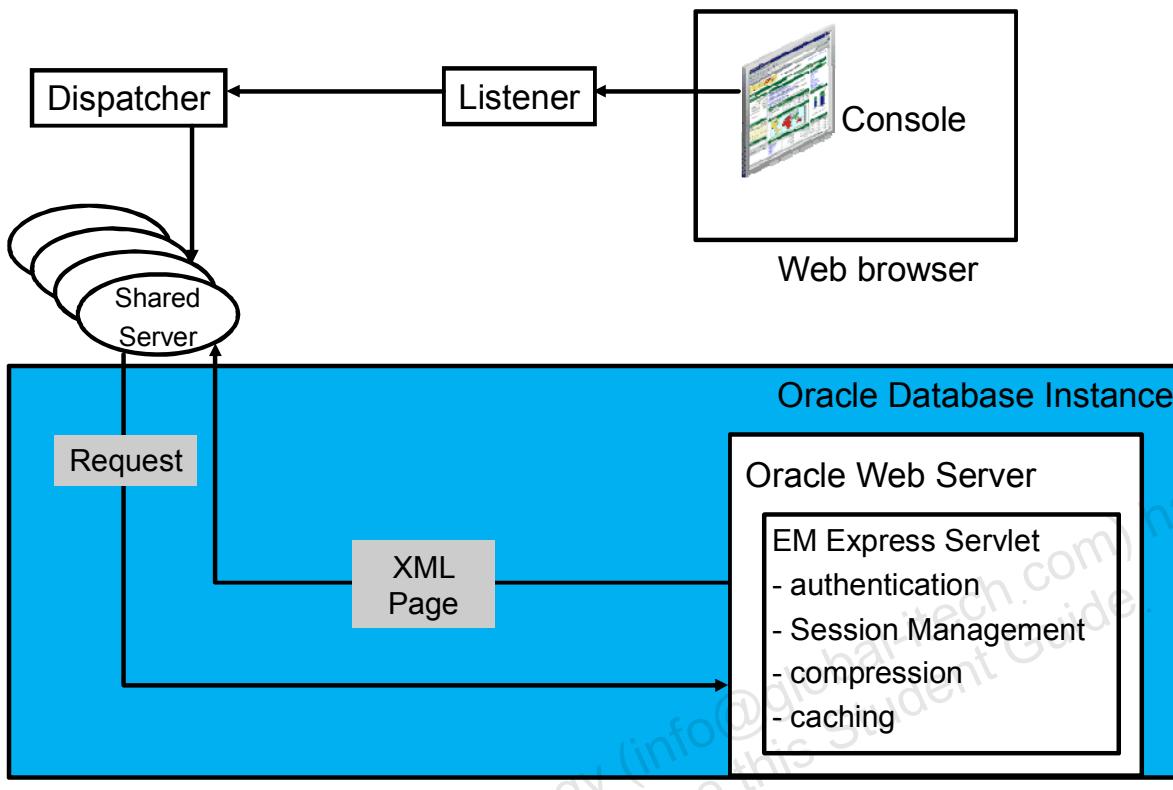
- a. Hosts
- b. Databases
- c. Application servers
- d. Web applications
- e. OMS and OMR
- f. All of the above



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

Answer: f

EM Database Express Architecture



ORACLE

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

Enterprise Manager (EM) Database Express is a lightweight administration tool. It provides an out-of-box web-based management solution for a single Oracle database (or database cluster), including performance monitoring, configuration management, administration, diagnostics and tuning. The goal of EM Database Express is to expose key Enterprise Manager Database features out-of-box with database installation, with low development cost and small footprint inside database. EM Database Express has a small footprint of 50 - 100 MB. It runs inside the database with minimal CPU and memory overhead because the database only runs SQL calls but the UI rendering is done in the browser itself.

EM Express uses a web-based console communicating with the built-in web server available in XML DB. As requests from the console are processed, the EM Express servlet handles the requests, including authentication, Session Management, compression, and caching. The servlet processes requests for reports and returns XML pages that are rendered by the web browser. Often a single request is made per page to reduce the number of round-trips to the database.

Configuring Enterprise Manager Database Express

Configure an HTTP or HTTPS listener port for each database instance:

- Verify DISPATCHERS parameter.

```
dispatchers=(PROTOCOL=TCP) (SERVICE=sampleXDB)
```

- Use DBMS_XDB_CONFIG.setHTTPSPort procedure.

```
exec DBMS_XDB_CONFIG.setHTTPSPort(5500)
```

- Connect to the Enterprise Manager Database Express console with:

```
https://hostname:5500/em
```

- Use a different port for each instance.
- Browser requires Flash Plugin.



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

With Oracle Database 12c, Enterprise Manager Database Control is no longer available. Enterprise Manager Database Express replaces it. Enterprise Manager Database Express is configurable with a single click in Database Configuration Assistant (DBCA).

Enterprise Manager Database Express requires that the XMLDB components are installed. All Oracle version 12.1.0 databases will have XMLDB installed.

To activate Enterprise Manager Database Express in a database verify that the DISPATCHERS initialization parameter has at least one dispatcher configured for the XMLDB service with the TCP protocol.

Use the setHTTPSPort (secured with SSL) or the setHTTPPPort procedure in the DBMS_XDB_CONFIG package to configure a used port on the server. Connect to the Enterprise Manager Database Express console with the URL shown in the slide if you configured the port using the setHTTPSPort procedure (default used by DBCA). Substitute the host name of the server, and the port number used in the setHTTPSPort procedure. If you configured the port using the setHTTPPPort procedure, use a URL like <http://hostname:5500/em>.

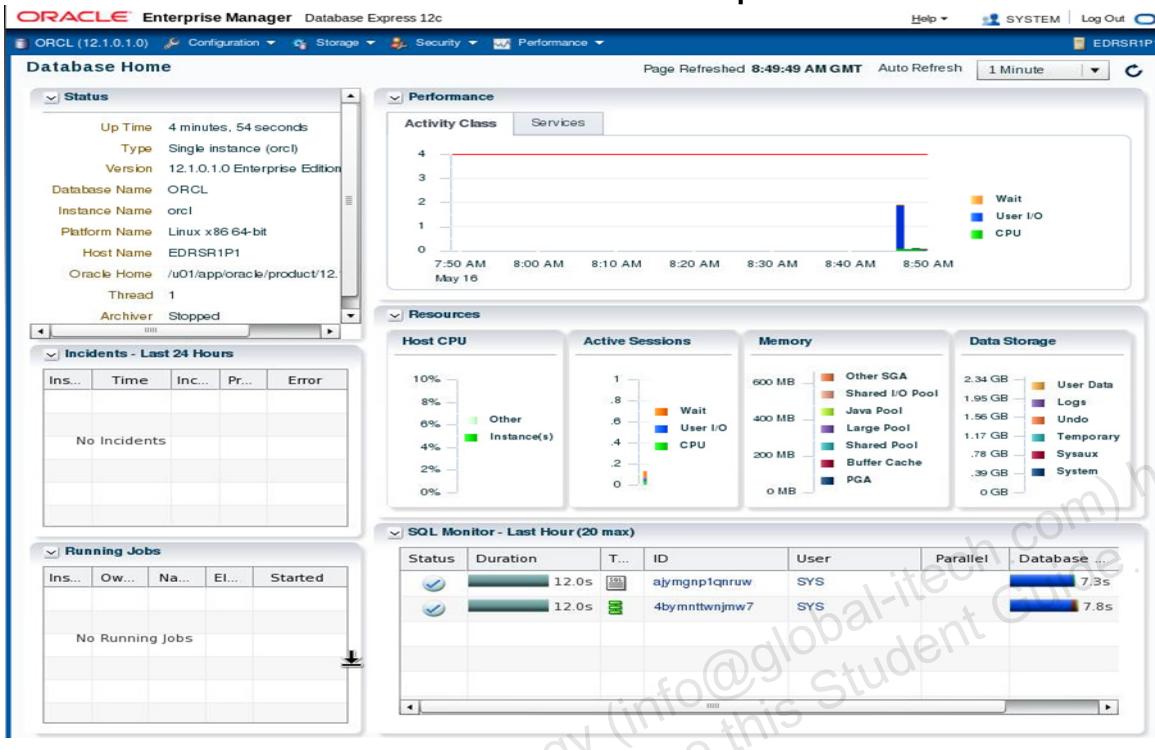
If you have multiple database instances to monitor on the same machine each one will require a different port. To find the port used for each database instance, use either of the following statements depending on the procedure used for the configuration:

```
SQL> SELECT dbms_xdb_config.gethttpsport FROM DUAL;
SQL> SELECT dbms_xdb_config.gethttpport FROM DUAL;
```

Enterprise Manager Database Express uses the Shockwave Flash (SWF) files, so the web browser must have the Flash plug-in installed.

Home Page

It is available when the database is open.



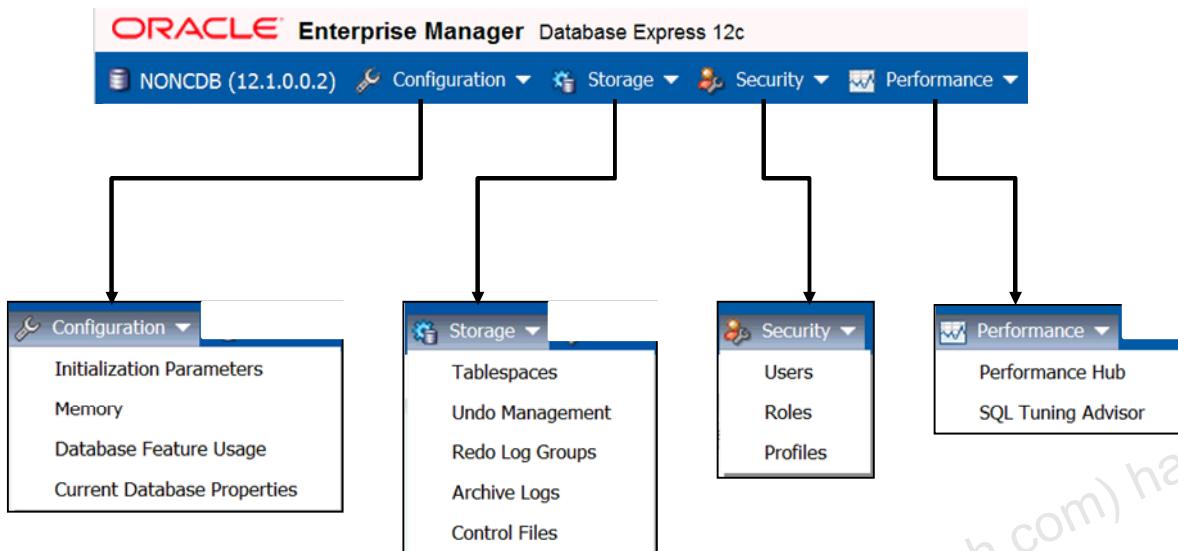
ORACLE

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

The Enterprise Manager Database Express Home page presents an overall view of the database instance status and activity.

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

Menus



ORACLE

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

The menu layout for Enterprise Manager Database Express is shown in the slide. There are four main menu items: Configuration, Storage, Security, and Performance. The menu selections for each of the main items are shown.

The Configuration menu includes: Initialization Parameters, Memory, Database Feature Usage, and Current Database Properties. The Storage menu includes Tablespaces, Undo Management, Redo Log Groups, Archive Logs, and Control Files. The Security menu includes: Users, Roles, and Profiles. The Performance menu includes: Performance Hub, and SQL Tuning Advisor.

In each of the menu areas, the menu selection directs you to a page that allows you to manage a particular area. For example the Configuration > Initialization Parameters selection displays a page that allows you to search, view and modify current and SPFILE initialization parameters. On many of the pages, when you select an action such as Create Tablespace, you get a pop-up dialog box that allows you to specify parameters, and then it creates a SQL command to perform the action. You may view the SQL command before you submit it, or you can copy and paste the SQL command.

Note: For a complete understanding of Oracle Enterprise Manager Database Express usage, refer to the following source of information:

- *Performance New Features for Oracle Database 12c Self-Study series*

Quiz

Enterprise Manager Database Express provides database administration functions for a:

- a. Single-instance database
- b. Cluster database instances

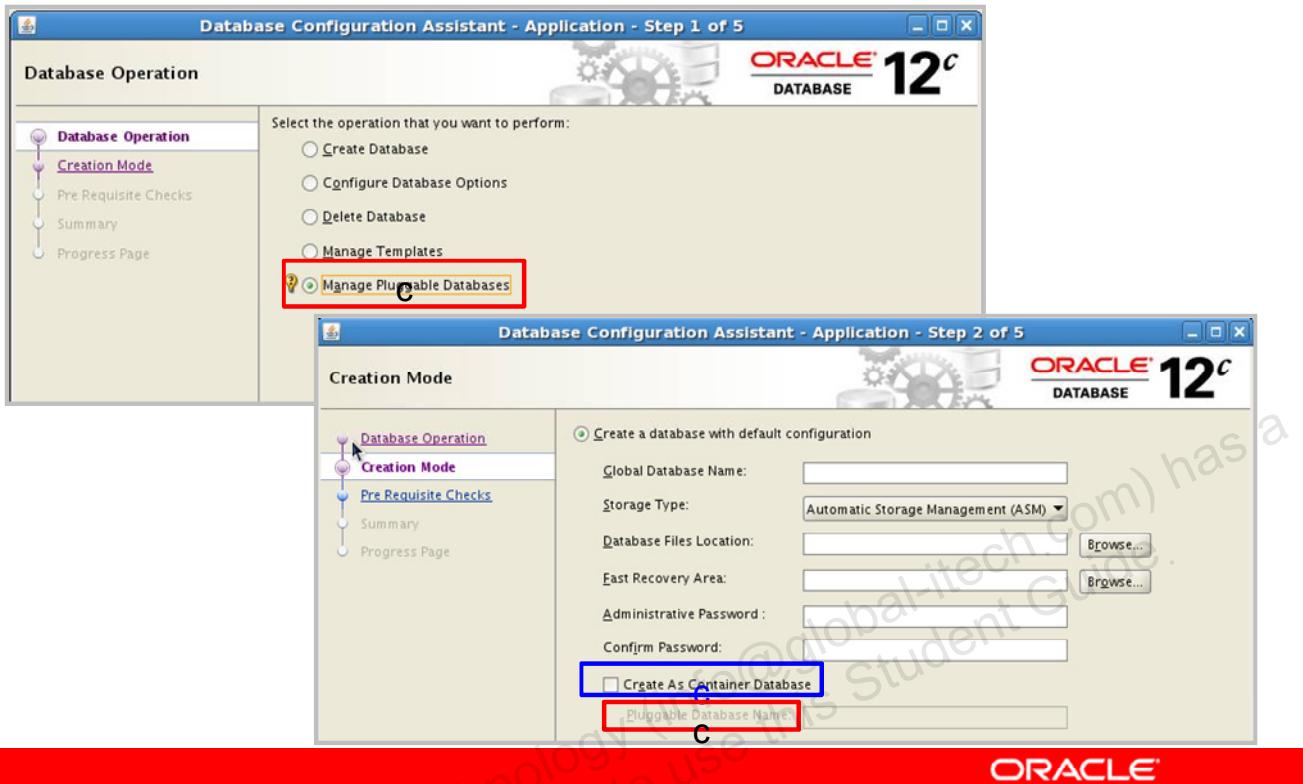


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Database Configuration Assistant

Multitenant container database and Pluggable database



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

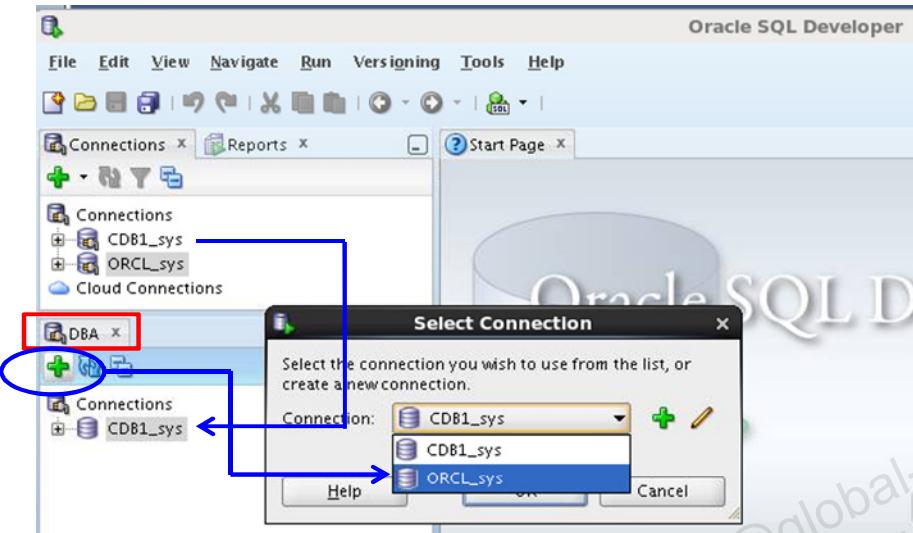
The Database Configuration Assistant in Oracle Database 12c allows the creation and management of new types of database:

- Multitenant container database
- Pluggable database

The creation and management of these new types of database are covered in the following module titled « Multitenant Container Databases and Pluggable Databases ».

Oracle SQL Developer: Connections

Perform DBA operations in the **DBA navigator** using **DBA connections**:



ORACLE

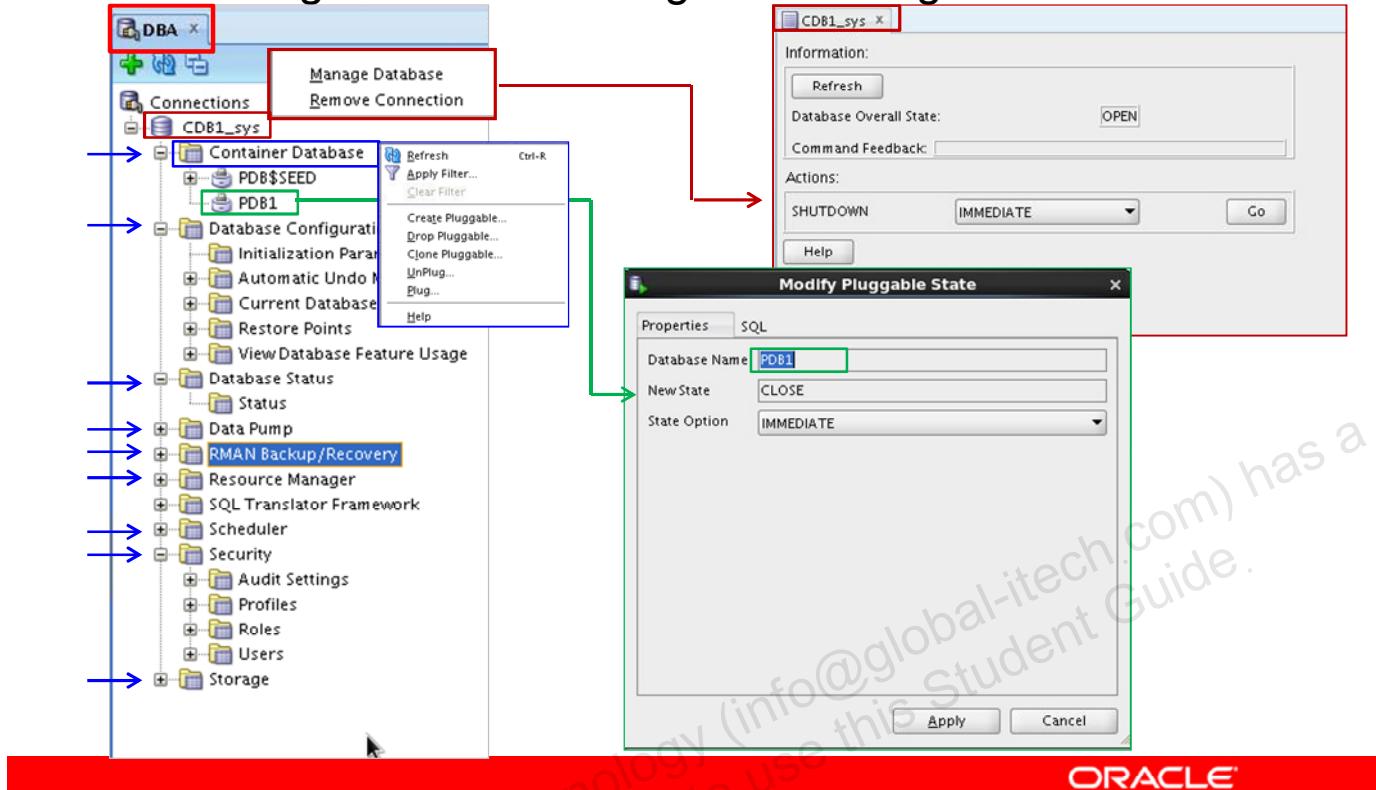
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer is a tool that allows stand-alone graphical browsing and development of database schema objects, as well as execution of database administrative tasks.

SQL Developer enables users with database administrator privileges to view and edit certain information relevant to DBAs and perform DBA operations. To perform DBA operations, use the DBA navigator, which is similar to the Connections navigator in that it has nodes for all defined database connections. If the DBA navigator is not visible, select View, then DBA. You should add only connections for which the associated database user has DBA privileges or at least privileges for the desired DBA navigator operations on the specified database.

Oracle SQL Developer: DBA Actions

Performing DBA tasks through **DBA** navigator:



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

ORACLE

The DBA operations that can be performed are the following:

- Pluggable database startup/shutdown
- Database configuration: Initialization Parameters, Automatic Undo Management, Current Database Properties, Restore Points, View Database Feature Usage
- Database status view
- Data Pump Export and Import jobs
- RMAN Backup/Recovery actions
- Resource Manager configuration
- Scheduler setting
- Security configuration like audit settings, profiles, roles, users
- Storage configuration for archive logs, control files, data files, redo log groups, tablespaces, temporary tablespace groups

Quiz

Oracle SQL Developer allows database administration operations:

- 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:

- Access and customize the Oracle Enterprise Manager Cloud Control interface
- Set named credentials
- Navigate in Oracle Enterprise Manager Database Express
- Use DBCA with new functionalities
- Use the SQL*Developer interface

Practice 1 Overview: Using Enterprise Manager Cloud Control

These practices cover the following topics:

- Accessing Enterprise Manager Cloud Control
- Setting the Summary page as the home page
- Adding a database instance as a new target monitored by EM CC
- Creating a new named credential
- Using the named credential



ORACLE

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

View the “Oracle Enterprise Manager 12c: Console Overview and Customization” demonstration (unless your instructor just demonstrated those topics) (8 mins).

Optional demonstrations about related topics are available:

- Oracle Enterprise Manager 12c: Create an Enterprise Manager Administrator (6 mins)
- Oracle Enterprise Manager 12c: Create and Use Named Credentials (6 mins)
- Oracle Enterprise Manager 12c: Create SSH Key Named Credentials (3 mins)
- Oracle Enterprise Manager 12c: Discover and Promote Unmanaged Hosts and Targets (3 mins)

Optional Oracle By Example (OBEs) about related topics are available:

- Oracle Enterprise Manager 12c Enterprise Ready Framework: Create and Use Credentials (60 mins)
- Oracle Enterprise Manager 12c Enterprise Ready Framework: Create a Super Administrator Account (5 mins)

Module

Multitenant Container Database and Pluggable Databases



ORACLE

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

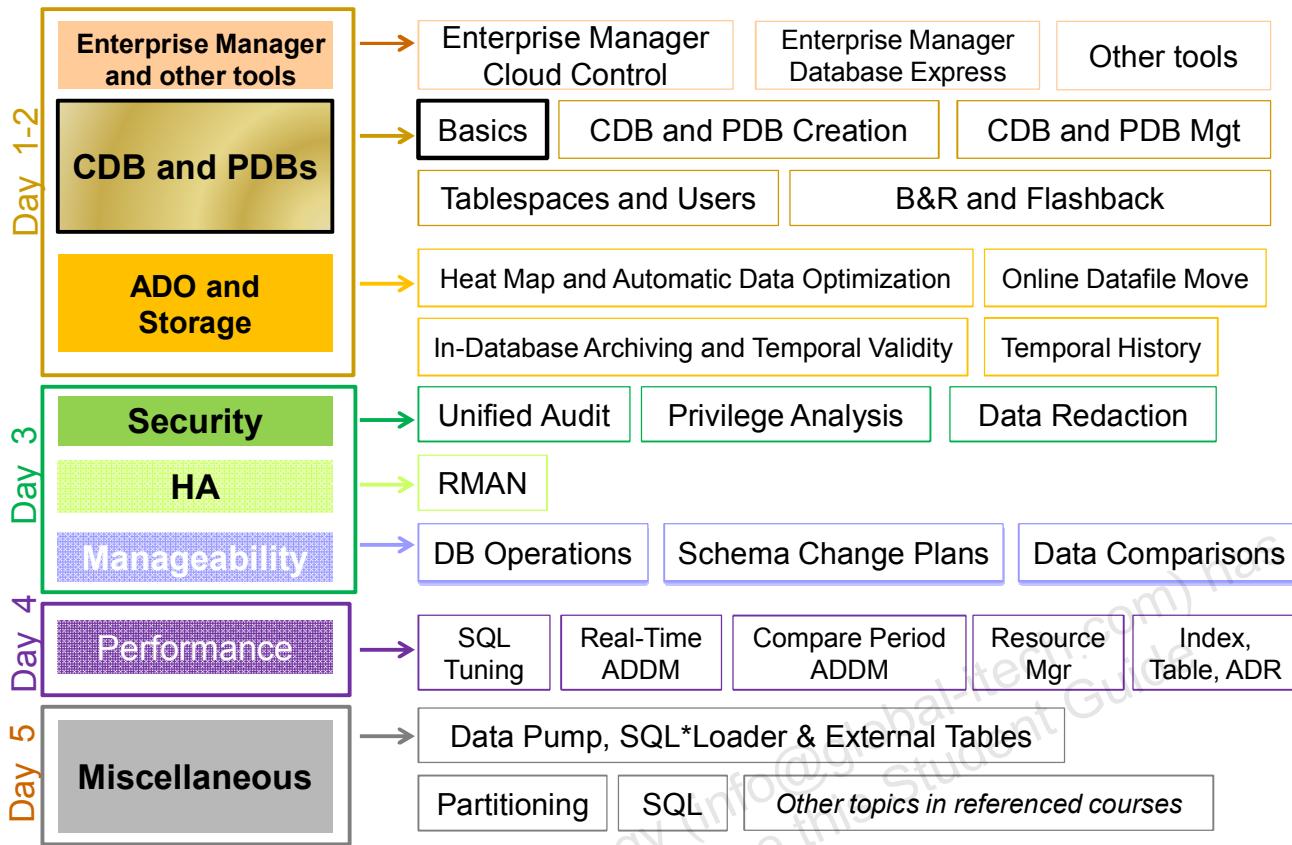
Basics of Multitenant Container Database and Pluggable Databases



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.

Database Consolidation: Oracle Multitenant is a new option in Oracle Database 12c. The multitenant architecture enables an Oracle database to contain a portable set of schemas, objects, and related structures that appears logically to an application as a separate database.

Objectives

After completing this lesson, you should be able to:

- Describe the multitenant architecture
- Describe the root and pluggable database containers
- Differentiate the root from a pluggable database
- Explain pluggable database plugging
- List impacts in various areas



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

Note

For a complete understanding of the multitenant architecture and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Administrator's Guide 12c Release 1 (12.1)*

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

- *Oracle Database 12c New Features Demo Series* demonstrations:
 - *Multitenant Architecture*
 - *Basics of CDB and PDB Architecture*
- *Oracle By Example (OBE)*:
 - *Performing Basic Tasks on Multitenant Container and Pluggable Databases*

Challenges

Many Oracle customers have large numbers of “departmental” applications built on Oracle RDBMS. They:

- Do NOT use a significant percentage of the hardware on which they are deployed
- Have instance and storage overhead preventing large numbers of “departmental” databases from being placed on the same physical and storage server
- Are NOT sufficiently complex to require 100% of the attention of a full time administrator
- Do require significant time to patch or upgrade all applications



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

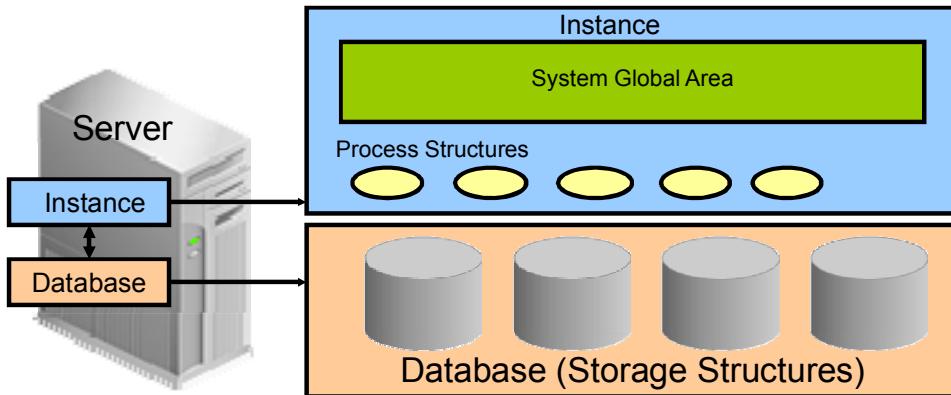
Oracle Database 12c brings a new architecture that lets you have many *pluggable databases* inside a single Oracle Database occurrence. What is the benefit of using the Oracle Multitenant new option in Oracle Database 12c?

Currently, many Oracle customers have large numbers of “departmental” applications built on Oracle RDBMS.

- These applications rarely use a significant percentage of the hardware on which they are deployed. A significant number of instances, and amount of storage allocation for all these small databases prevent these from being placed on the same physical and storage server.
- Moreover, they are typically not sufficiently complex to require 100% of the attention of a full time administrator.
- To better exploit hardware and DBA resources, customers would prefer to have most of these departmental applications consolidated onto a single Oracle RDBMS deployment.

The Oracle Multitenant option allows DBAs to consolidate large numbers of small departmental database applications into a single, larger RDBMS installation.

Oracle Database in 11g Release 2



- Multiple monolithic or non-CDBs share nothing:
 - Too many background processes
 - High shared/process memory
 - Many copies of Oracle metadata

ORACLE

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

In Oracle 11g database, the only kind of database that is supported is a non-CDB. We refer to the old architecture as the non-CDB architecture—the term *non-CDB* will be used as a shorthand for an occurrence of a pre-12.1 database that uses the pre-12.1 architecture—it requires its own instance and, therefore, its own background processes, memory allocation for the SGA, and needs to store the Oracle metadata in its data dictionary.

When you have to administer small departmental database applications, you have to create as many databases as applications and therefore multiply the number of instances, consequently the number of background processes, memory allocation for the SGAs, and provision enough storage for all data dictionaries of these databases.

When you need to upgrade your applications to a new version, you have to upgrade each database which is time-consuming for the DBA.

New Multitenant Architecture: Benefits

- Operates **multiple databases in a centrally managed platform** to lower costs:
 - Less instance overhead
 - Less storage cost
- Reduces DBA resources costs and maintains security
 - No application changes
 - **Fast and easy provisioning**
 - **Time saving for patching and upgrade**
 - **Separation of duties** between:
 - Different application administrators
 - Application administrators and DBA
 - Users within application
- **Provides isolation**



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

Consolidating many non-CDB databases onto a single platform reduces instance overhead, avoids redundant copies of data dictionaries, and consequently storage allocation, and benefits from fast provisioning, time saving upgrading, better security through separation of duties and application isolation. The new 12c database that consolidates databases together is a multitenant container database or CDB, and a database consolidated within a CDB, a pluggable database or PDB.

DBA resource costs are reduced with:

- *No application change and very fast provisioning:* A new database can be provisioned very quickly. A clone of a populated database can be created very quickly. A populated database can be quickly unplugged from its CDB on one platform and quickly plugged into a CDB on a different platform. A non-CDB can quickly be plugged into a CDB.
- *Fast upgrade and patching of the Oracle Database version:* The cost (time taken and human effort needed) to upgrade many PDBs is the cost of upgrading a single Oracle Database occurrence. You can also upgrade a single PDB by unplugging it and plugging it into a CDB at a different Oracle database version.

The multitenant architecture maintains:

- *Secure separation of duties*: The administrator of an application can do all the required tasks by connecting to the particular PDB that implements its back end. However, someone who connects to a PDB cannot see other PDBs. To manage PDBs as entities (for example, to create or drop or unplug or plug one), the system administrator needs to connect to the CDB. For these specific tasks, new privileges need to be granted.
- *Isolation of applications* that may not be achieved manually unless using Database Vault for example. A good example of isolation is dictionary separation enabling Oracle Database to manage the multiple PDBs separately from each other and from the CDB itself.

Other Benefits of Multitenant Architecture

- Ensures **full backwards-compatibility** with non-CDBs
- Fully operates with RAC
- Is integrated with Enterprise Manager and Resource Manager
- Allows central management and administration of multiple databases
 - Backups / disaster recovery
 - Patching and upgrades

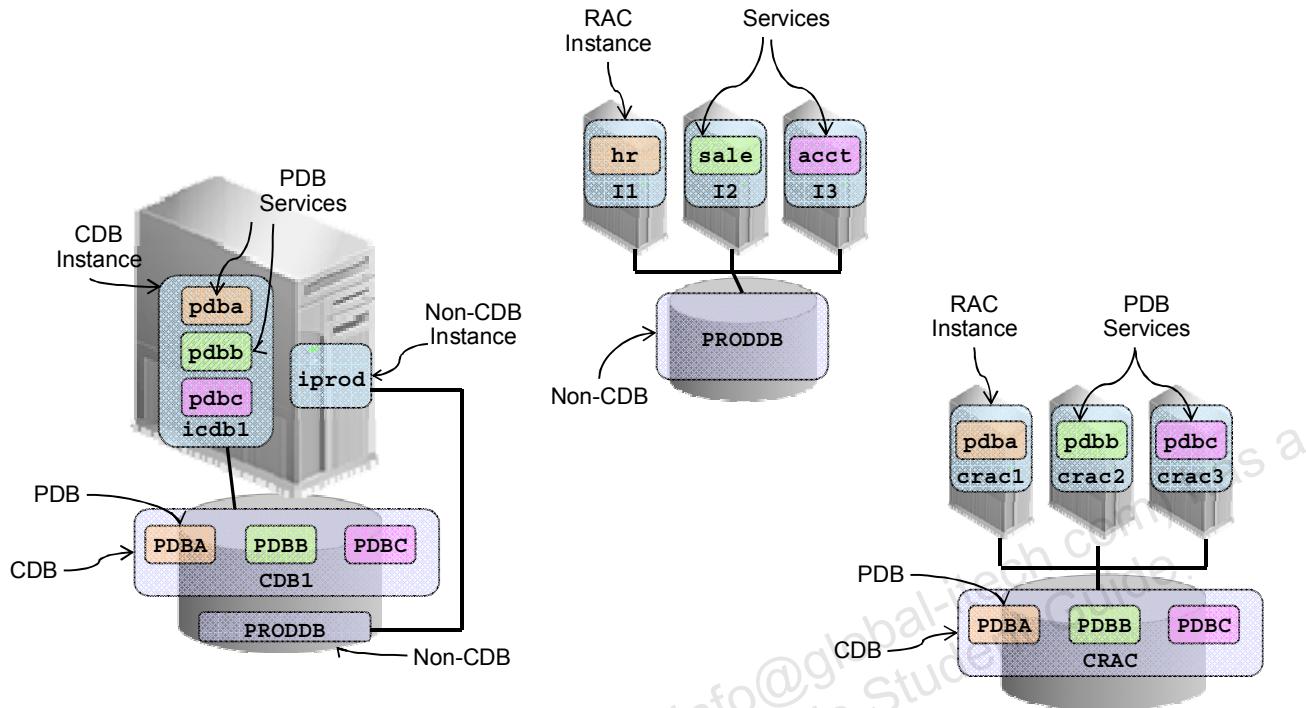


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

- The multitenant architecture ensures the backwards-compatibility principle. An example is the data dictionary views. The DBA_OBJECTS view shows the same results in a PDB as in a non-CDB for a particular application.
- The multitenant architecture is designed to be fully interoperable with RAC. Each instance in a RAC opens the CDB as a whole. A session sees only the single PDB it connects to.
- Enterprise Manager integrates CDBs and models the separation of duties of the CDB administrator and the PDB administrator.
 - A CDB can be defined as a target. An Enterprise Manager user can be given the credentials to act as a CDB administrator in such a target.
 - A PDB can be set up as a subtarget of a CDB target. An Enterprise Manager user can be given the credentials to act as a PDB administrator in such a target. An Enterprise Manager user that has been set up with the credentials to act as a PDB administrator for a particular PDB is able to connect to that one PDB and is unaware of the existence of peer PDBs in the same CDB. Moreover, when the intention is to carry out the duties of an application administrator, this Enterprise Manager user is unaware that the environment is a CDB and not a non-CDB.

- Resource Manager is extended with new between-PDB capabilities to allow the management of resources between the PDBs within a CDB. The backward-compatibility principle implies that Resource Manager must function in exactly the same way within a PDB as it does in a non-CDB.
- When you upgrade a whole CDB with n PDBs, you achieve the effect of upgrading n non-CDBs for the cost of upgrading one non-CDB.

Configurations



ORACLE

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

Which are the possible instance/database configurations in Oracle Database 12c?

In Oracle Database 11g, each database instance is associated with one and only one database. In a RAC environment, several instances can be associated to a database.

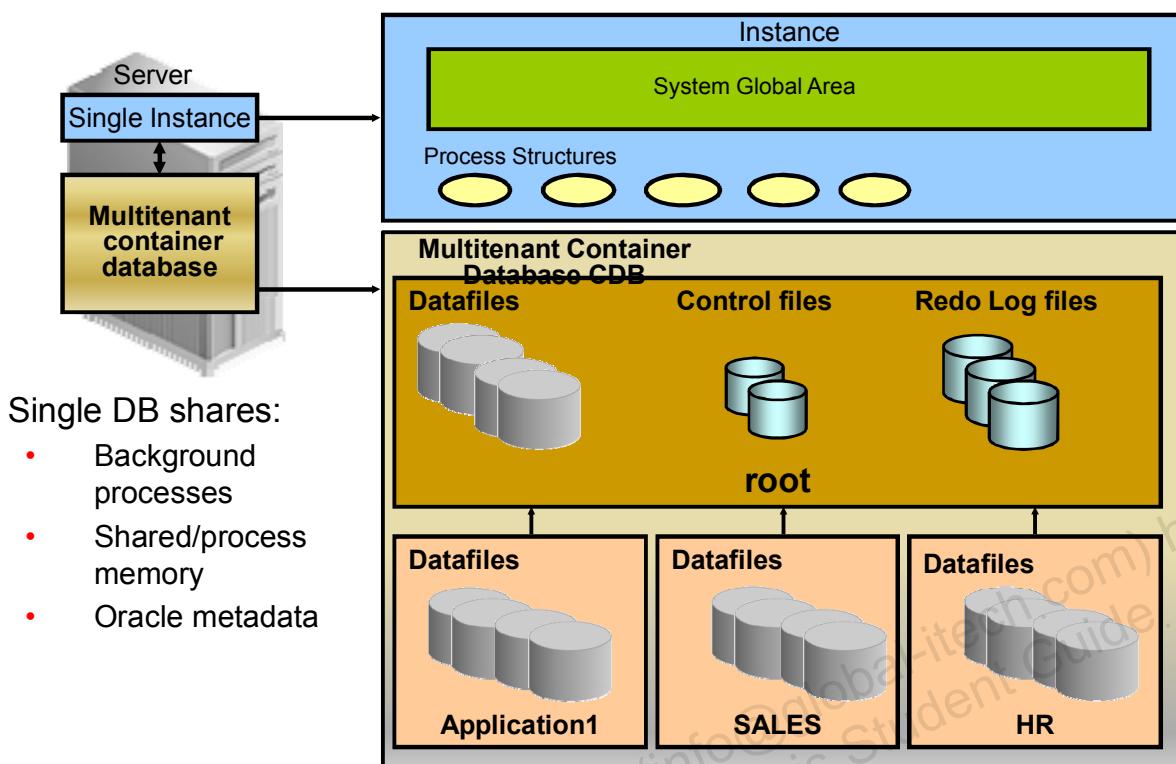
In Oracle Database 12c, an instance is associated with an entire CDB.

If there are multiple databases on the same server, then there is a separate and distinct instance for each non-CDB or CDB. An instance cannot be shared between a non-CDB and CDB.

In Oracle Database 12c, there are three possible configuration options:

- **Multitenant configuration:** typically more than one PDB per CDB, but can hold zero, one or many PDBs at any one time, taking advantage of the full capabilities of the new architecture, which requires the licensed Oracle Multitenant option
- **Single-tenant configuration:** the special case of the new architecture which does not require the licensed option
- **Non-CDB:** the old Oracle Database 11g architecture

Multitenant Container Database



ORACLE

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

The slide is an example of the consolidation of three applications that were deployed into three distinct non-CDBs into a single one. The graphic in the slide shows a multitenant container database with four containers: the root, and three pluggable databases. Each pluggable database has its own dedicated application, and is managed either by its own DBA or by the container administrator that is **SYS** user of the root container, a common user. This common **SYS** user can manage the root container and every pluggable database.

A pluggable database is a set of database schemas that appears logically to users and applications as a separate database. But at the physical level, the multitenant container database has a database instance and database files, just as a non-CDB does. Nothing changes: neither the client code, nor the database objects.

It is easy to plug non-CDBs in to a CDB. A CDB avoids redundancy of:

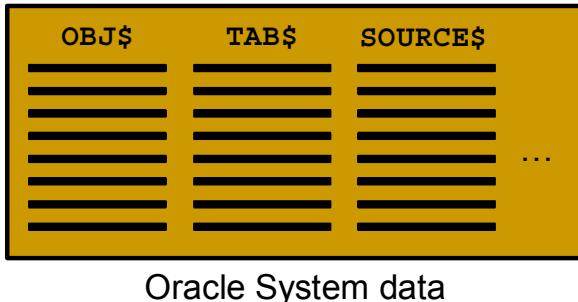
- Background processes
- Memory allocation
- Oracle metadata in several data dictionaries

A CDB grouping several applications ends up with one instance, consequently one set of background processes, one SGA allocation and one data dictionary in the root container, common for all PDBs, each PDB maintaining its own application data dictionary.

When applications need to be patched or upgraded, the maintenance operation is performed only once on the CDB and consequently all applications are updated at the same time.

Pristine Installation

After the initial database creation, the only objects are Oracle-supplied objects.



ORACLE

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

How is Oracle metadata shared between several applications in a non-CDB?

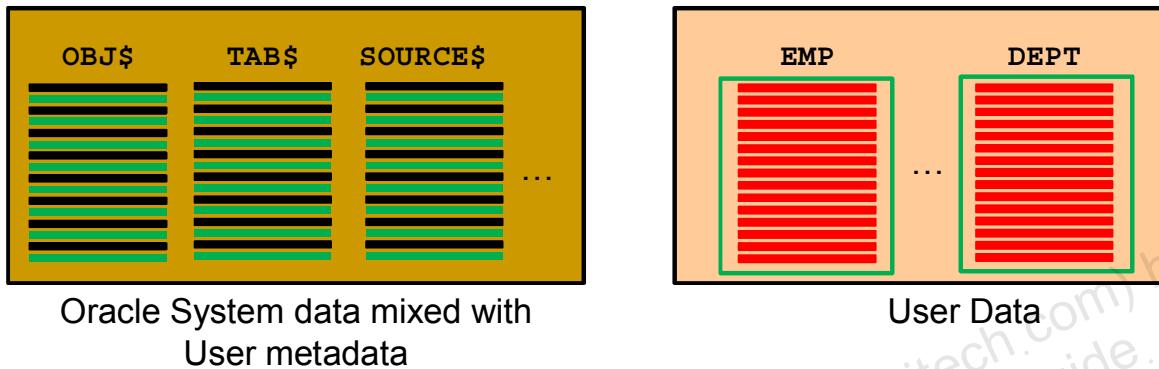
Immediately after the creation of the Oracle database, the only objects in the data dictionary are the Oracle-supplied objects. At this point, there is no user data.

The only schemas in the database are those required by the database system.

Adding User Data

In a non-CDB, user data is added:

- The metadata is mixed with the Oracle supplied data in the data dictionary.



ORACLE

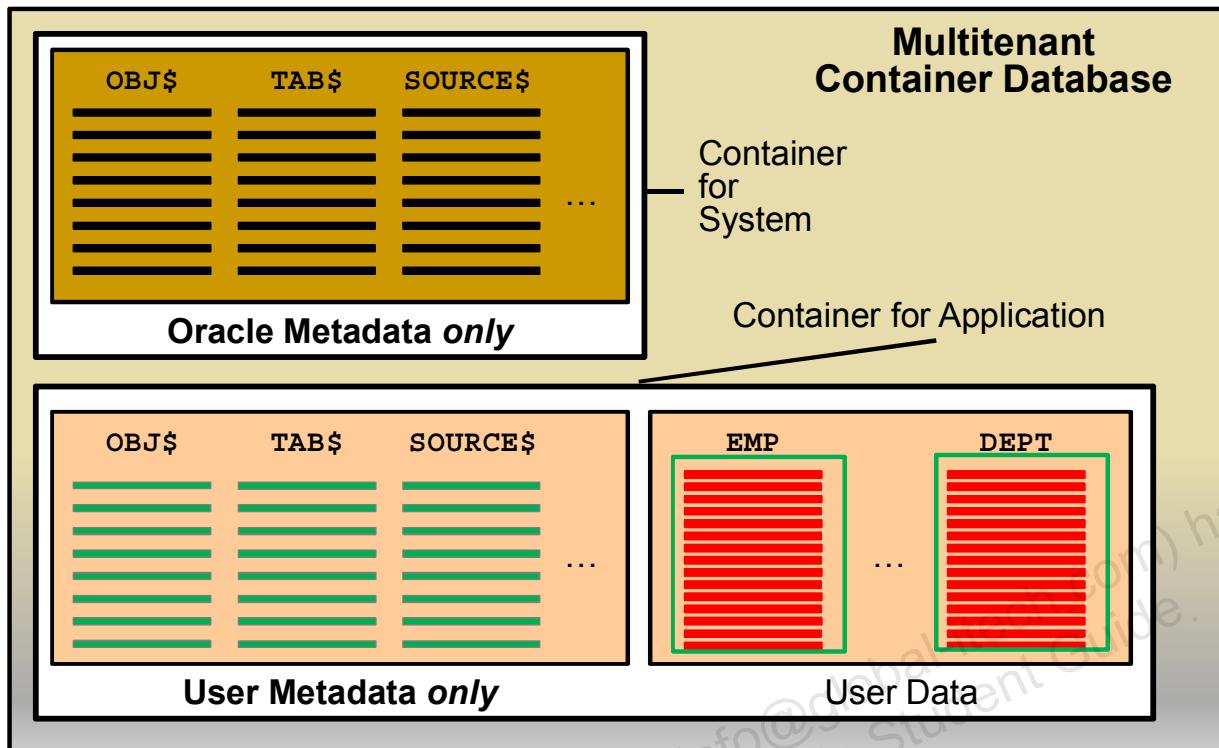
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the non-CDB, users and user data are added.

Best practice is to add this data in tablespaces dedicated to user data. Storing the data in separate tablespaces helps to protect, secure, and transport the data more easily. The metadata, though, is all mixed together in the data dictionary:

- Object definitions
- User definitions
- PL/SQL code
- Other user-created objects

Separating SYSTEM and User Data



ORACLE

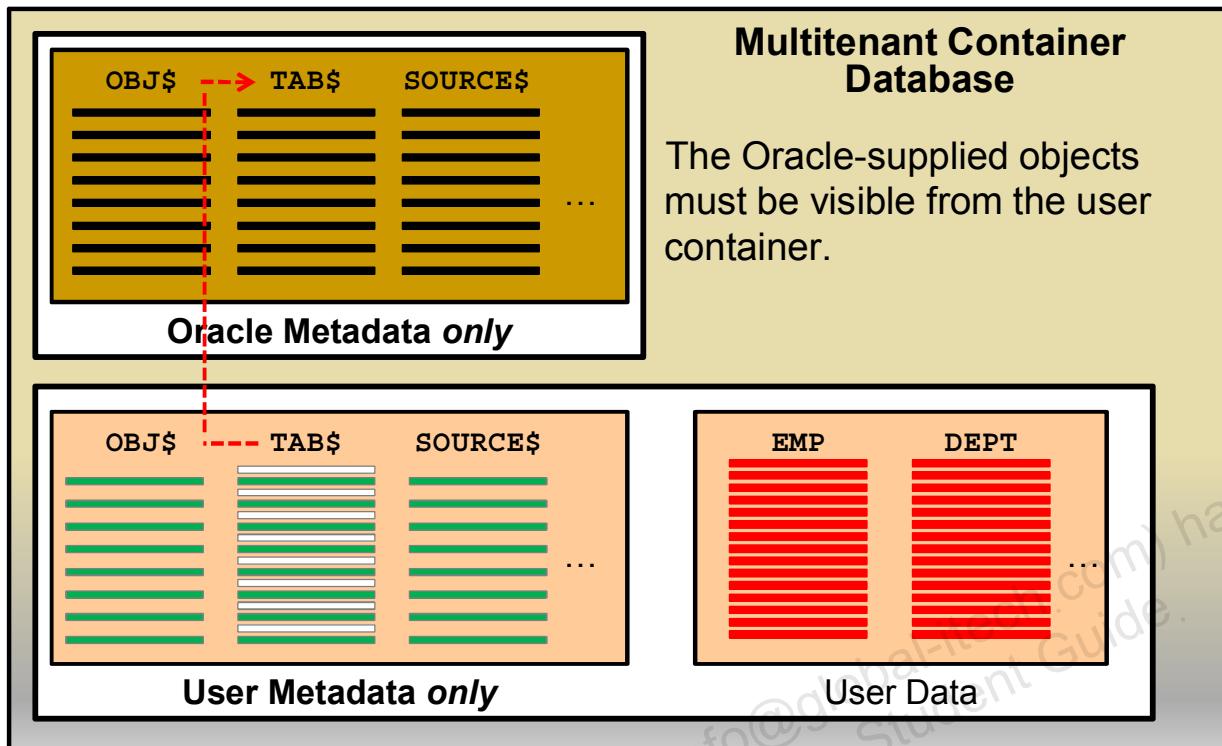
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In Oracle 12c, the concept of containers is introduced to separate the Oracle-supplied objects and the user data, including the metadata, into distinct containers.

This can be thought of as a horizontal partitioning. There is a SYSTEM tablespace in each container holding a data dictionary.

- There is a dictionary in the Oracle metadata-only container that has the metadata for the Oracle-supplied objects.
- There is a dictionary in the user container holding the user metadata.

SYSTEM Objects in the USER Container



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

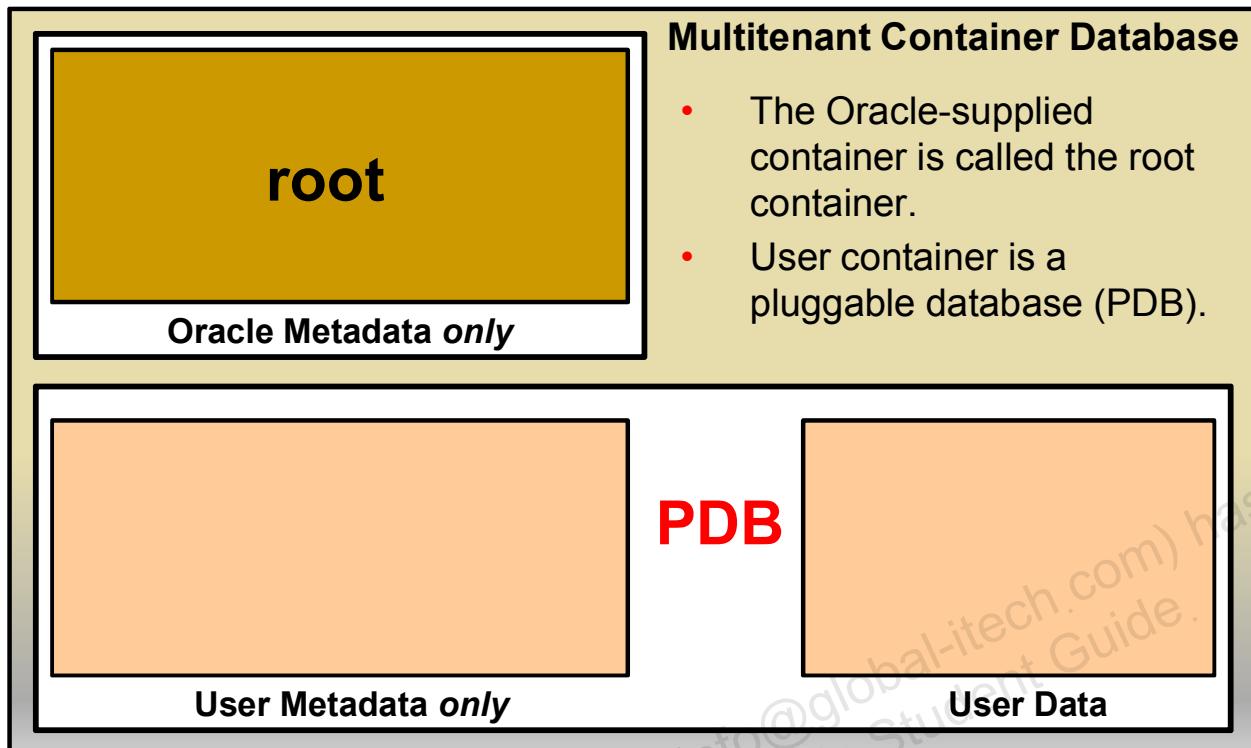
One of the goals of the multitenant architecture is that each container has a one-to-one relationship with an application.

Separating the metadata is the first step, the second is allowing the application or users inside the “user” container to access the Oracle-supplied objects.

The Oracle objects could have been duplicated in each user container, but that takes a lot of space, and would require every user container to be upgraded each time an Oracle-supplied object changes, for example, with patches.

A simple solution is to provide pointers from the user container to the Oracle-supplied objects to allow these “system” objects to be accessed without duplicating them in the user container. The user container has the pieces it needs to be a complete environment for a database application. The application can run in the container just as it does in a non-CDB.

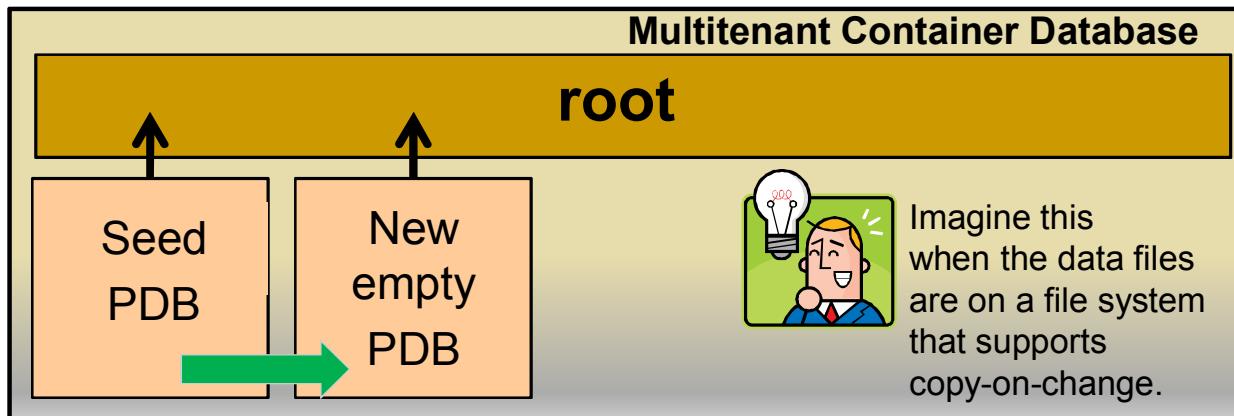
Naming the Containers



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

The Oracle-supplied objects reside in a container called the root container (named CDB\$ROOT). The user container is called a pluggable database (PDB) and has the name you give it when creating it or plugging it into the CDB.

Provisioning a Pluggable Database



Four methods:

- Create new PDB from `PDB$SEED` pluggable database.
- Plug in a non-CDB.
- Clone a PDB from another PDB into the same or another CDB.
- Plug an unplugged PDB into another CDB.

ORACLE

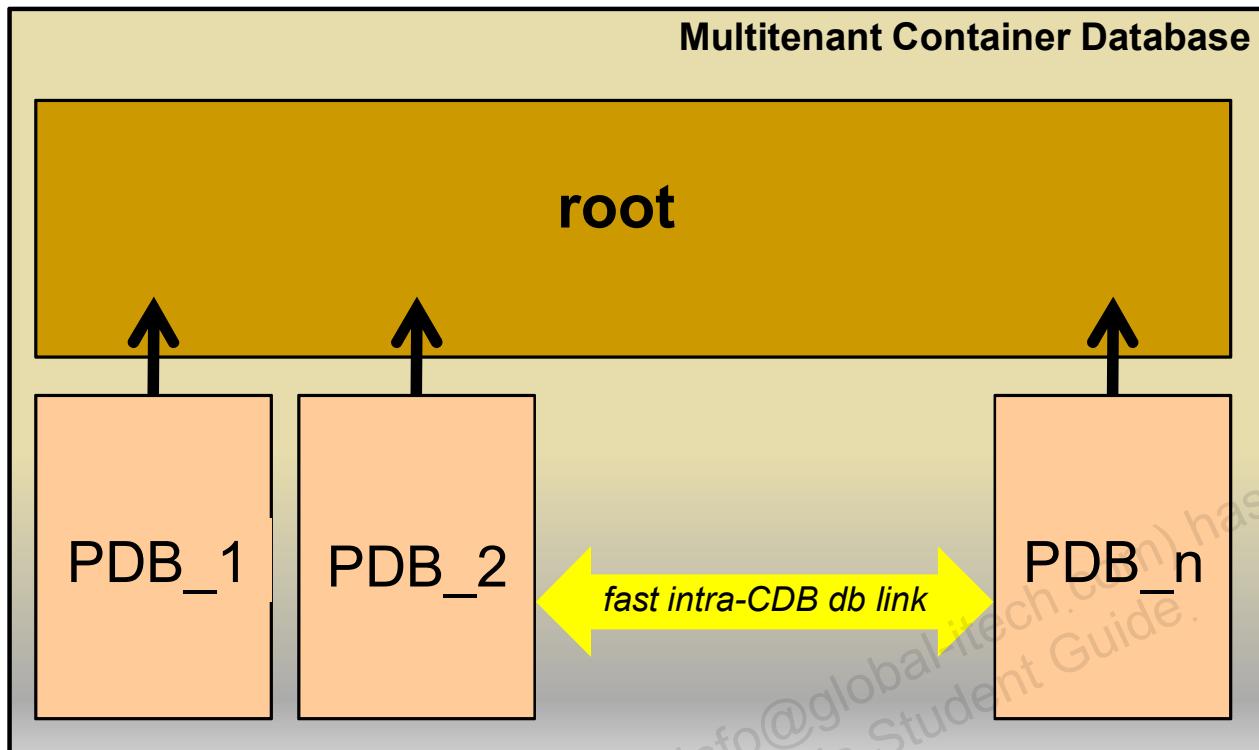
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

To create a new PDB, use the provided seed PDB. This seed container is named `PDB$SEED` and is a part of every CDB. When you create a new PDB, the seed PDB is cloned and gives the new PDB the name you specify. This operation is very fast. It is measured in seconds. The time is mostly to copy the files.

There are four methods to provision pluggable databases:

- Create a new PDB from `PDB$SEED` pluggable database: for example for a brand new application implementation.
- Create a new PDB from a non-CDB: plug the non-CDBs in a CDB as PDBs, as part of migration strategy. It is also a good way to consolidate the non-CDBs into a CDB.
- Clone a PDB from another PDB into the same or another CDB: an example of this method is application testing.
- Plug an unplugged PDB into another CDB, for example, instead of upgrading a multitenant container database from one release to another, you can unplug a pluggable database from one Oracle Database release, and then plug it in to a newly created multitenant container database from a higher release.

Interacting Within Multitenant Container Database



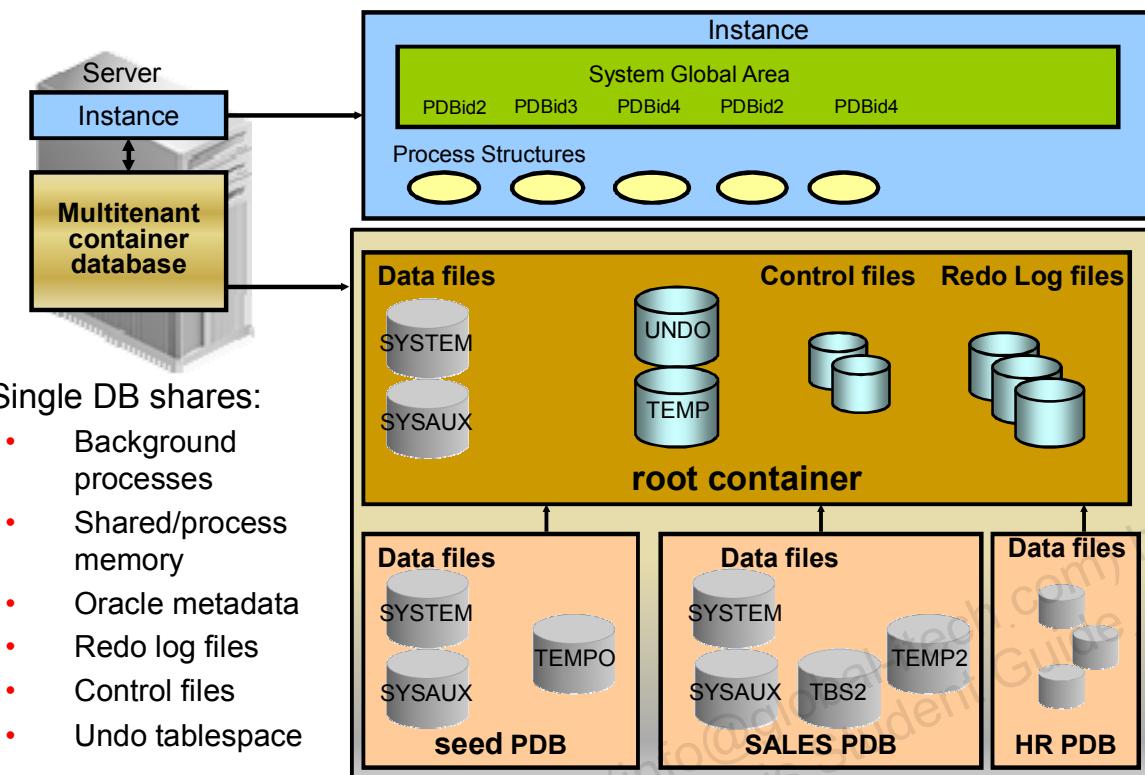
ORACLE

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

With a multitenant architecture that holds several PDBs, these once separated non-CDBs now may reside in a single instance, sharing memory, disk and CPU resources, but maintaining application separation.

These databases shared data using database links. The database link still works, but now because the “link” communication does not leave the instance, the link is very fast.

Multitenant Container Database Architecture



ORACLE

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

The graphic in the slide shows a CDB with four containers: the root, the seed, and two PDBs.

The two applications use a single instance and are maintained separately.

At the physical level, the CDB has a database instance and database files, just as a non-CDB does.

- The redo log files are common for the whole CDB. The information it contains is annotated with the identity of the PDB where a change occurs. Oracle GoldenGate is enhanced to understand the format of the redo log for a CDB. All PDBs in a CDB share the ARCHIVELOG mode of the CDB.
- The control files are common for the whole CDB. The control files are updated to reflect any additional tablespace and data files of plugged PDBs.
- The UNDO tablespace is common for all containers.
- A temporary tablespace common to all containers is required. But each PDB can hold its own temporary tablespace for its own local users.
- Each container has its own data dictionary stored in its proper SYSTEM tablespace, containing its own metadata, and a SYSAUX tablespace.
- The PDBs can create tablespaces within the PDB according to application needs.
- Each data file is associated with a specific container, named *CON_ID*.

Containers

Two types of containers in V\$CONTAINERS:

- The root container
 - The first container created at CDB creation
 - Mandatory
 - Oracle system-supplied common objects and metadata
 - Oracle system-supplied common users and roles
- Pluggable database containers (PDBs)
 - A container for an application:
 - Tablespaces (permanent and temporary)
 - Schemas / Objects / Privileges
 - Created / cloned / unplugged / plugged
 - Particular seed PDB
 - PDB\$SEED provides fast provisioning of a new PDB
 - Limit of 253 PDBs in a CDB including the seed
 - Limit of 512 services in a CDB



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

To summarize, a CDB is an Oracle database that contains the root and eventually several pluggable databases.

What is a pluggable database (PDB) in a CDB? A PDB is the lower part of the horizontally partitioned data dictionary plus the user's quota-consuming data.

A non-CDB cannot contain PDBs.

The multitenant architecture enables an Oracle database to contain a portable collection of schemas, schema objects, and nonschema objects that appears to an Oracle Net client as a separate database. For the PDBs to exist and work, the CDB requires a particular type of container, the root container, generated at the creation of the CDB. The root is a system-supplied container that stores common users, which are users that can connect to multiple containers, and system-supplied metadata and data. For example, the source code for system-supplied PL/SQL packages is stored in the root. When the root container exists, you can create containers of the other type, the PDB.

There is only one seed PDB in a CDB. The seed PDB is a system-supplied template that is used to create new PDBs.

A CDB can contain up to 253 PDBs including the seed. 252 user-defined PDBs can therefore be created.

The V\$CONTAINERS view display all containers including the root.

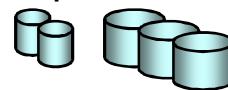


Questions: Root Versus PDBs

CDBA

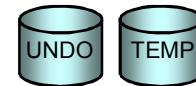
What belongs to the CDB and not to a specific container?

- Control files and redo log files



What is in the root that is not in PDBs?

- UNDO and default TEMP tablespace



- System supplied metadata

	NAME	TYPE
TAB\$	2	
USER\$	2	

- Shared Oracle-supplied data

- PL/SQL Oracle-supplied packages
(DBMS_SQL ...)

- PDBs service names

Table SYS.SERVICE\$

NAME
PDB_SALES
PDB_HR

- CDB dictionary views providing information across PDBs

Views CDB_xxx

TABLE_NAME	CON_ID
EMPLOYEES	1
TEST	2

- CDB RM plan

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

What belongs to the CDB and not to a specific container?

- Control files and redo log files

What is in the root that does not exist in PDBs?

- Shared UNDO and default database temporary tablespace
- Oracle-supplied metadata
- Shared Oracle-supplied data
- CDB views providing information across PDBs
- CDB resource manager plan allowing resource management between PDBs within a CDB

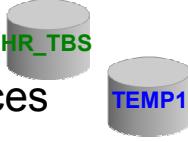
Note: A multitenant container database administrator (CDBA) is responsible for administering the CDB.



Questions: PDBs Versus Root

What is in a PDB that is not in the root nor in another PDB?

PDBA

- Application tablespaces 
- Local temporary tablespaces
- Local users   and local roles  
 - Local users connect to the PDB where they exist
- Non-shared local metadata

NAME	TYPE
EMPLOYEES	2
JOB\$	2
- Non-shared application data with other PDBs

EMP_NAME
SMITH
JOHN
- PDB RM plan

ORACLE

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

What is in a PDB that is not in the root?

1. Application tablespaces
2. Local temporary tablespaces
3. Local users and local roles
4. Non-shared local metadata
5. PDB resource manager plan allowing resource management within PDB

Note: A pluggable database administrator (PDBA) is responsible for administering its own PDB.

Terminology

- Common versus Local:
 - Users
 - Roles
 - Privileges
- CDB versus PDB level:
 - CDB Resource Manager plan versus PDB RM plan
 - Unified audit at CDB or PDB level
 - XStream at CDB or PDB level



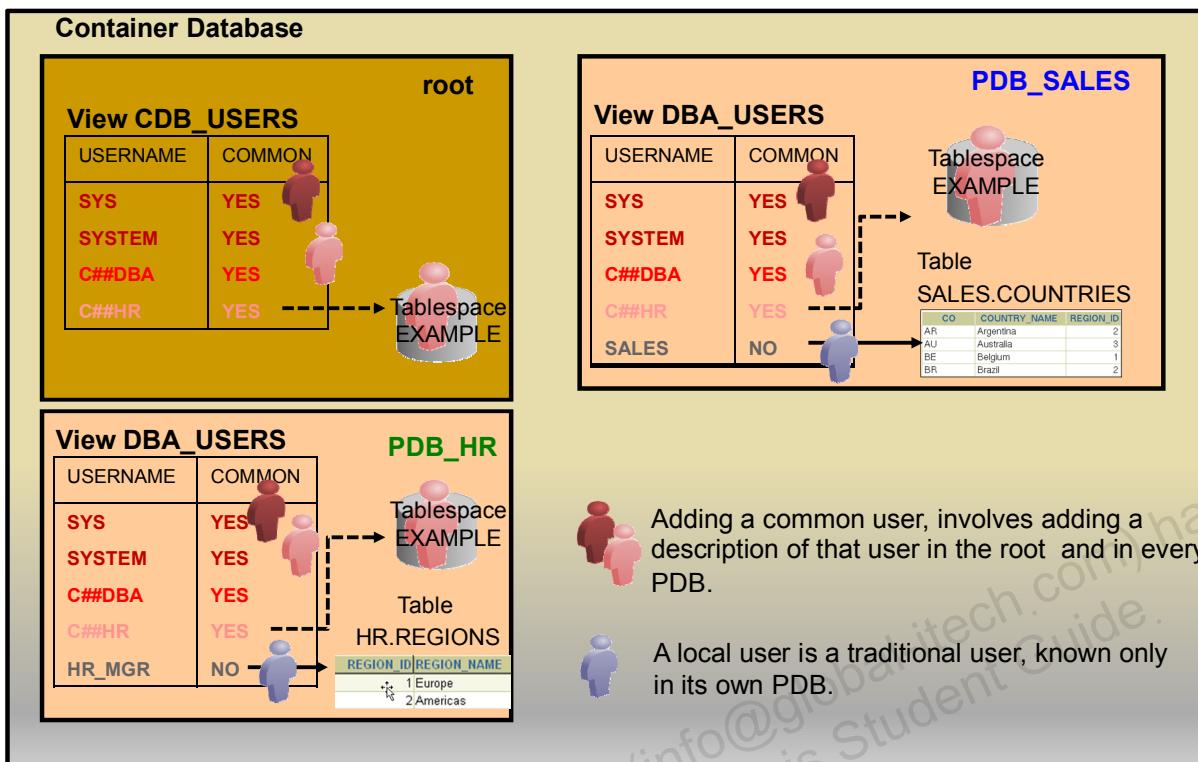
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There is a new terminology for new entities: common versus local entities, such as:

- Common users versus local users
- Common roles versus local roles
- Common privileges versus local privileges, privileges granted commonly for all containers versus privileges granted locally within a PDB
- CDB resource management versus PDB resource management. CDB resource management works at CDB level, and PDB resource management at PDB level.
- XStream Out is only available at CDB level and XStream In only at PDB level. XStream consists of Oracle Database components and application programming interfaces (APIs) that enable client applications to receive data changes from an Oracle database and send data changes to an Oracle database. These data changes can be shared between Oracle databases and other systems. The other systems include non-Oracle databases, non-RDBMS Oracle products, file systems, third party software applications.

XStream consists of two major features: XStream Out and XStream In. XStream Out provides Oracle Database components and APIs that enable you to share data changes made to an Oracle database with other systems. XStream In provides Oracle Database components and APIs that enable you to share data changes made to other systems with an Oracle database.

Common and Local Users



ORACLE

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

Local User

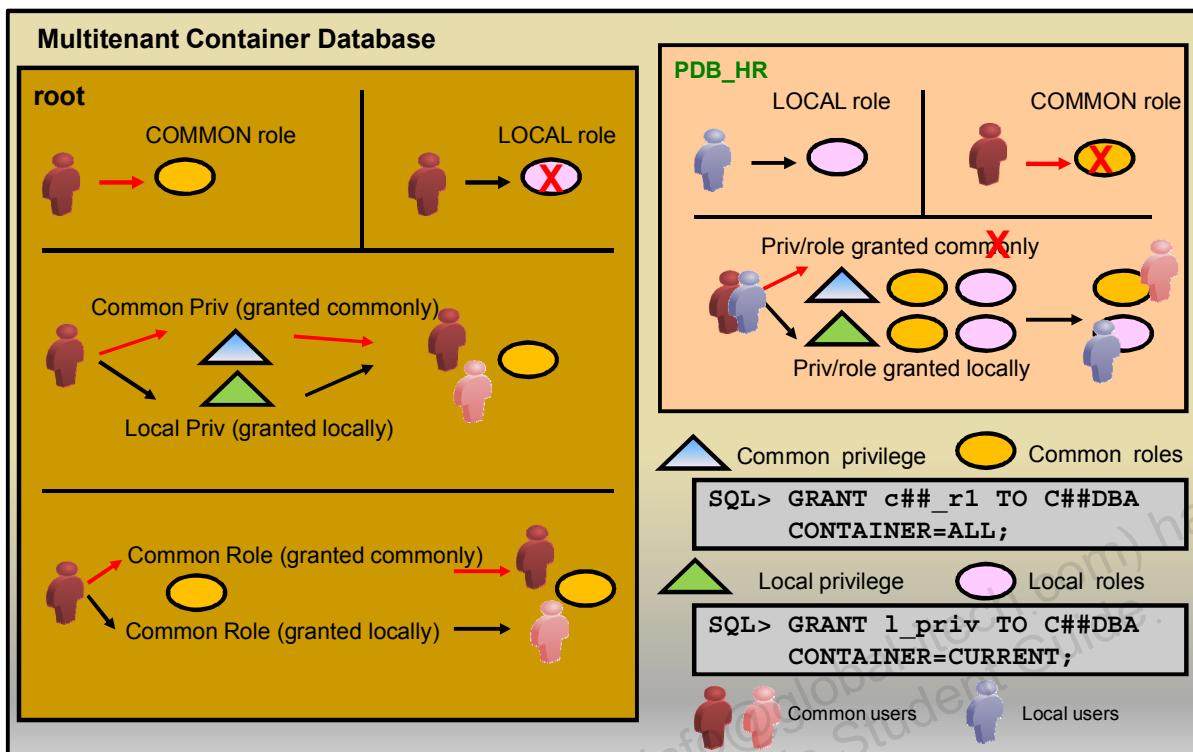
A user in a non-CDB maps to a *local user* in a PDB.

- A local user is defined in the PDB's own data dictionary—and so is not known outside of that PDB.
- A local user can connect only to the PDB where it is defined.
- A local user is specific to a particular PDB and owns a schema in this PDB.
- According to the privileges granted, a user can work on the application data within the PDB or with other PDBs' application using database links. Moreover, there cannot be any local users defined in the root.

Common User

- A *common user* is defined in the root's data dictionary.
- Only common users can be defined in the root: Creating a common user allows the CDB administrator to create at once a user that is replicated in each PDB.
- A common user is known, not only where it is defined in the root, but also in every PDB that belongs to the CDB.
- A common user can perform administrative tasks specific to the root or PDBs, such as plugging and unplugging PDBs, starting up the CDB, or opening a PDB when granted the proper privileges.

Common and Local Privileges and Roles



ORACLE

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

Local Roles

A role in a non-CDB maps to a *local role* in a PDB. A local role is defined in the PDB's own data dictionary—and so it is not known outside of that PDB and can only be used within its PDB.

Common Roles

Besides the local role, you can create common roles that are defined in every container. This way, it is easy to create at once a role that is replicated in all PDBs. It is at the creation time that you specify the nature of the role: local or common. Common roles as well as common users can only be created in the root by common users. Moreover, there cannot be any local roles defined in the root. All Oracle-supplied predefined roles are common roles.

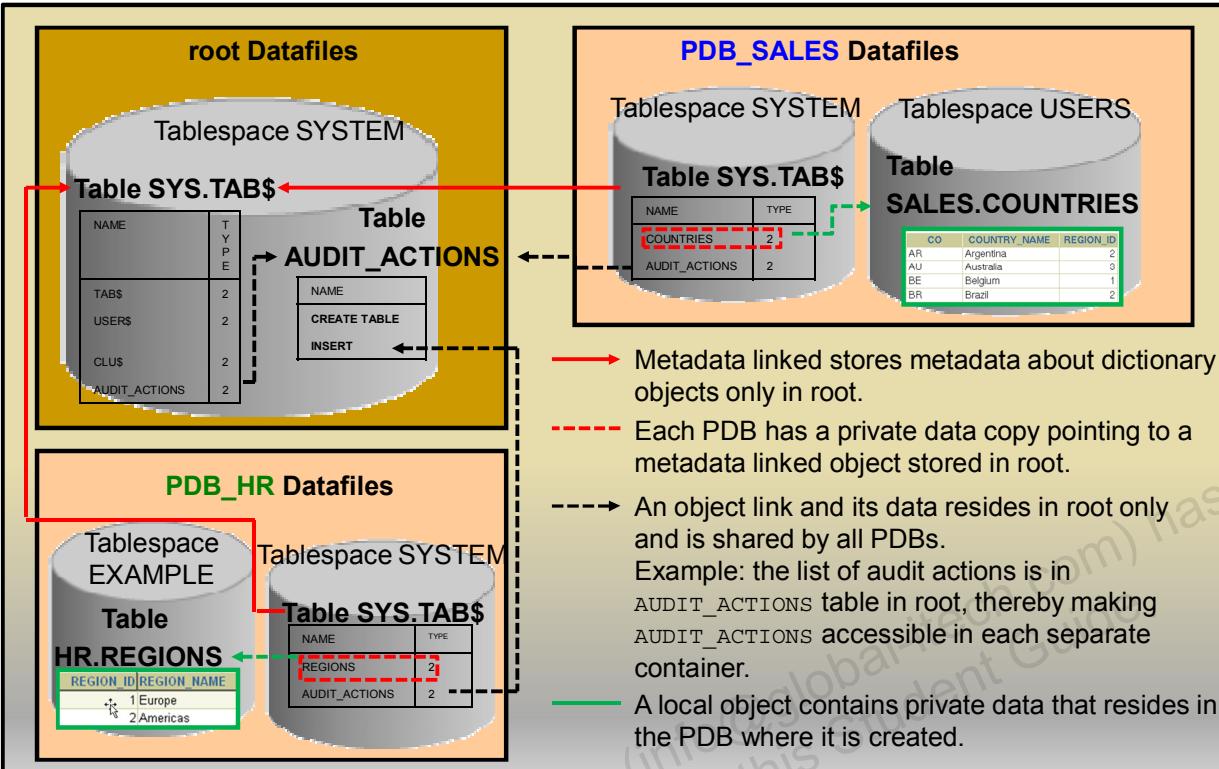
Local and Common Privileges

The privileges are commonly referred to as local or common privileges, but to be more precise a privilege is either granted locally with the clause CONTAINER=CURRENT or commonly with the clause CONTAINER=ALL.

The same rule applies to roles: common roles can be granted commonly or locally to common users or roles. Common roles may contain privileges that apply across the CDB, that is, commonly granted for all containers, and can also contain locally granted privileges that apply only to an individual PDB, whereas local roles do not contain any commonly granted privileges.

Shared and Non-Shared Objects

Multitenant Container Database



ORACLE

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

In a non-CDB, all objects are local objects. Common objects are being introduced in 12c to avoid having to store redundant representations of data and metadata across a CDB and to simplify the process of upgrading a CDB. Shared objects exist only in Oracle-supplied schemas.

A shared object may be metadata-linked or object-linked.

- Metadata-linked objects store metadata about dictionary objects only in the root. Each PDB has a private data copy of an object pointing to a metadata link stored in the root.
- An object-linked object and its data resides in the root only and is shared by all PDBs.

Example

The list of audit actions is in `SYS.AUDIT_ACTIONS` table in the root, thereby making `AUDIT_ACTIONS` accessible in each separate container.

Data Dictionary Views

CDB_XXX All objects in the multitenant container database across all PDBs

DBA_XXX All of the objects in a container or pluggable database

ALL_XXX Objects accessible by the current user

USER_XXX Objects owned by the current user

```
SQL> SELECT view_name FROM dba_views WHERE view_name like 'CDB%';
```

- CDB_pdbs: All PDBS within CDB
- CDB_tablespaces : All tablespaces within CDB
- CDB_users : All users within CDB (common and local)

DBA dictionary views providing information within PDB:

```
SQL> SELECT table_name FROM dict WHERE table_name like 'DBA%';
```

ORACLE

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

For backwards-compatibility, DBA views show the same results in a PDB as in a non-CDB: DBA_OBJECTS shows the objects that exist in the PDB from which you run the query. This implies, in turn, that though the PDB and the root have separate data dictionaries, each data dictionary view in a PDB shows results fetched from both of these data dictionaries. The DBA_XXX views in the root shows, even in a populated CDB, only the Oracle-supplied system, as is seen in a freshly-created non-CDB. This is another advantage of the new architecture. To support the duties of the CDB administrator, a new family of data dictionary views is supported with names like CDB_XXX. Each DBA_XXX view has a CDB_XXX view counterpart with an extra column, Con_ID, that shows from which container the listed facts originate. Query the CDB_XXX views from the root and from any PDB. The CDB_XXX views are useful when queried from the root because the results from a particular CDB_XXX view are the union of the results from the DBA_XXX view counterpart over the root and all currently open PDBs. When a CDB_XXX view is queried from a PDB, it shows only the information that it shows in its DBA_XXX view counterpart. If you connect to the root and query CDB_USERS, you get the list of users, common and local, of each container. Now if you query DBA_USERS, you get the list of common users (you know that in the root, only common users exist). Now if you connect to a PDB, and query CDB_USERS or DBA_USERS, you get the same list of users, common and local, of the PDB.

The same backward-compatibility principle applies also to each of the familiar v\$views.

Impacts

- One character set for all PDBs (Unicode recommended)
- PDB initialization parameters but a single SPFILE
- No PDB qualified database object names
 - ~~SELECT * FROM HR.apps.tab1~~
 - Use DB Links: SELECT * FROM **apps.tab1@HR**
- Oracle Data Guard at CDB level
- Oracle Database Vault per PDB only
- One master key per PDB to encrypt PDB data
- Unified audit both at CDB and PDB level
- Oracle Scheduler
- Oracle GoldenGate
- Oracle Streams
- Oracle XStream both at CDB and PDB level



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

- There is only one character set and one single spfile for the CDB. The different parameter values of each PDB are stored in a dictionary table.
- There is no PDB qualified object name. To access an object in another PDB, use a DB link.
- Oracle Data Guard works exactly the same in a CDB as it does in non-CDBs. This feature is not to be available on a per-PDB basis. Open of a physical standby CDB will always open the root in READ ONLY mode and this also means that no PDB may be open in a mode other than READ ONLY.
- In Oracle Database Vault, each PDB has its own Database Vault metadata. Database Vault constructs like realms are isolated within a PDB.
- Each PDB has its own master key used to encrypt data in the PDB. The master key must be transported from the source database wallet to the target database wallet when a PDB is moved from one host to another. For column encryption, each PDB maintains its own ENC\$ which is not a metadata-linked object.

- A unified audit configuration is visible and enforced across all PDBs. This provides the ability to create audit policies used by all PDBs and audit policies used exclusively for each PDB. An audit configuration that is not enforced across all PDBs means it applies only within a PDB and is not visible outside it. A unified audit framework enables administrators to avoid configuring auditing separately for each container.
- In general, all scheduler objects created by the user can be exported/imported into the PDB using data pump. Predefined scheduler objects will not get exported, and that means that any changes made to these objects by the user will have to be made once again when the database has been imported into the pluggable database. However, this is how import/export works currently. A job defined in a PDB will run only if a PDB is open.
- XStream is a programmatic interface to allow a client application access to the changes in the database, known as XStream Outbound Server. XStream Inbound Server allows a client application to feed changes into the database and takes advantage of the apply process available in the database. Oracle GoldenGate is the logical replication and XStream is licensed via the Oracle GoldenGate license. Capturing changes from the database must always be from the root, due to the single unified redo log for the CDB. The XStream outbound can be configured to capture changes from a PDB or the entire CDB. Applying changes via Oracle GoldenGate is done per PDB. An XStream inbound server is configured to apply changes into a specific PDB and performs all of its work within the context of the PDB.
- Oracle Streams continues to be supported in the Oracle Database 12c non-CDBs but is no longer enhanced for new database features or data types. Oracle Streams cannot be used with CDBs.

Quiz

Which of the following are true?

- a. Oracle-supplied metadata resides only in the root container.
- b. The seed PDB can sometimes be opened for very particular operations.
- c. A PDB can have the same name in different CDBs.



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

Answer: a

Quiz

Which of the following are true? There is:

- a. Only one SYSTEM tablespace per CDB
- b. Only one instance per PDB
- c. A set of redo log files per PDB
- d. Only one UNDO tablespace per CDB
- e. One SYSAUX tablespace per PDB



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

Answer: d, e

Quiz

You can create common users in a PDB.

- a. True
- b. False



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

Answer: b

Summary

In this lesson, you should have learned how to:

- Describe the multitenant architecture
- Describe the root and pluggable database containers
- Differentiate the root from a pluggable database
- Explain pluggable database plugging
- List impacts in various areas

Practice 2 Overview: Exploring a Multitenant Container Database

These practices cover the following topics:

- Exploring the CDB processes and files
- Displaying CDB_xxx views

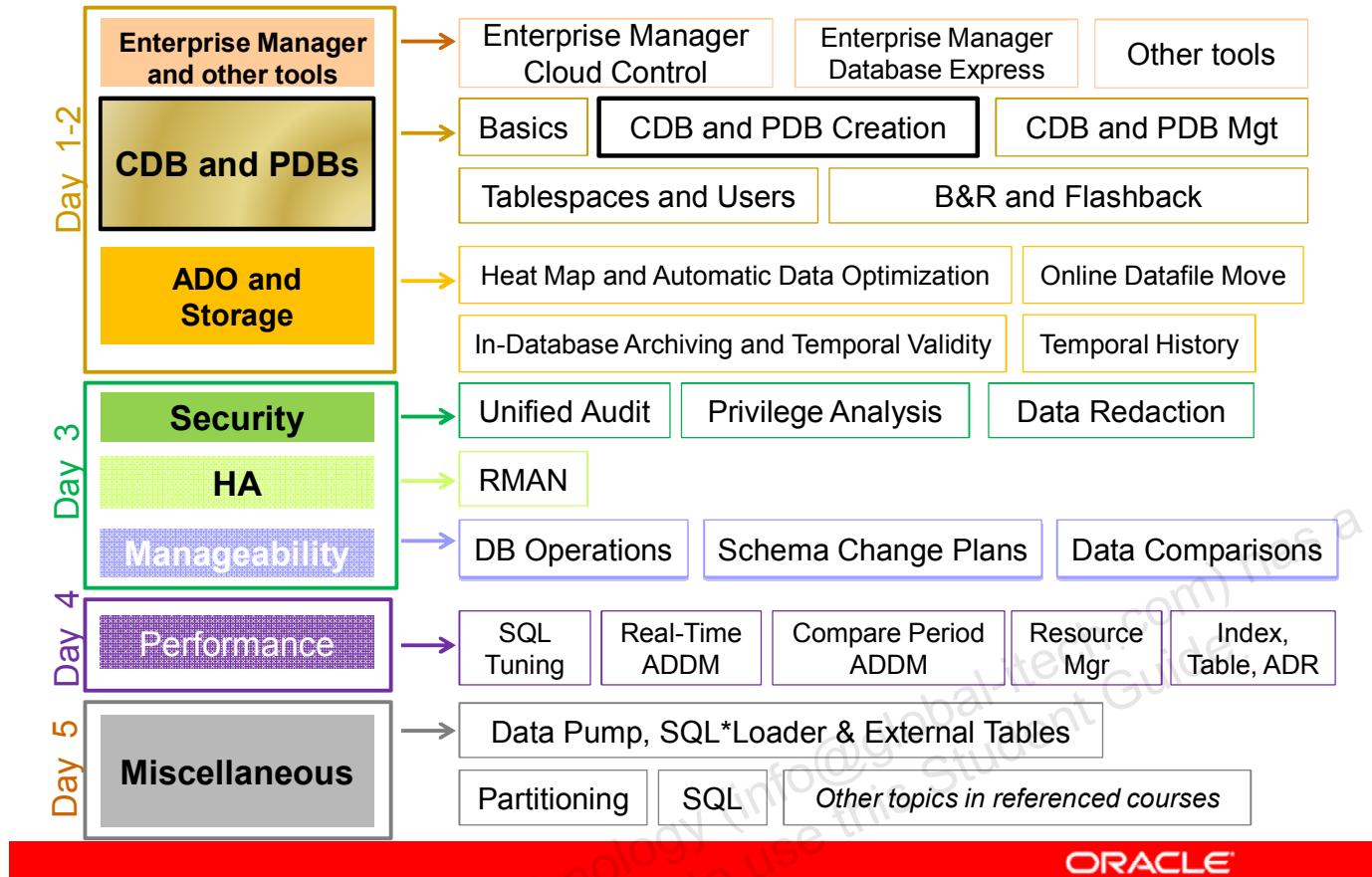
Creating Multitenant Container Databases and Pluggable Databases



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.

Objectives

After completing this lesson, you should be able to:

- Configure and create a CDB
- Create a PDB from PDB\$SEED
- Create a PDB from a non-CDB
- Clone a PDB into the same CDB
- Unplug and plug a PDB from one CDB to another a CDB
- Explore the instance
- Explore the structure of PDBs
- Drop a PDB
- Migrate a pre-12.1 non-CDB database to CDB



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

Note

For a complete understanding of the 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*

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

- *Oracle Database 12c New Features Demo Series* demonstrations :
 - *How to Create and Drop CDBs*
 - *How to Create and Drop PDBs - Create a PDB from seed*
 - *How to Create and Drop PDBs - Clone a PDB from another PDB*
 - *How to Create and Drop PDBs - Plug non-CDB into a CDB as a PDB*
 - *How to Create and Drop PDBs - Unplug a PDB and plug the PDB in another CDB*
 - *How to Create and Drop PDBs - Drop a PDB*
- *Oracle By Example (OBE)*:
 - *Unplugging and Plugging Pluggable Databases*
 - *Cloning Pluggable Databases*
 - *Plugging a non-CDB as a Pluggable Database into a Multitenant Container Database*

Goals

Create a multitenant container database:

- To consolidate many pre-12.1 non-CDBs into a single, larger database
- To prepare a container
 - For plugging any future new application
 - For testing applications
 - For diagnosing application performance
- To simplify and reduce time for patching and upgrade



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

Why create multitenant container databases?

There are several reasons for creating a multitenant container database as explained in the slide.

Tools

	SQL*Plus	OUI	DBCA	EM Cloud Control	SQL Developer	DBUA
Create a new CDB or PDB	Yes	Yes	Yes	Yes (PDB only)	Yes (PDB only)	
Explore CDB instance, architecture, PDBs	Yes			Yes	Yes	
Upgrade a 12.1 CDB to 12.x CDB	Yes			Yes		Yes



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

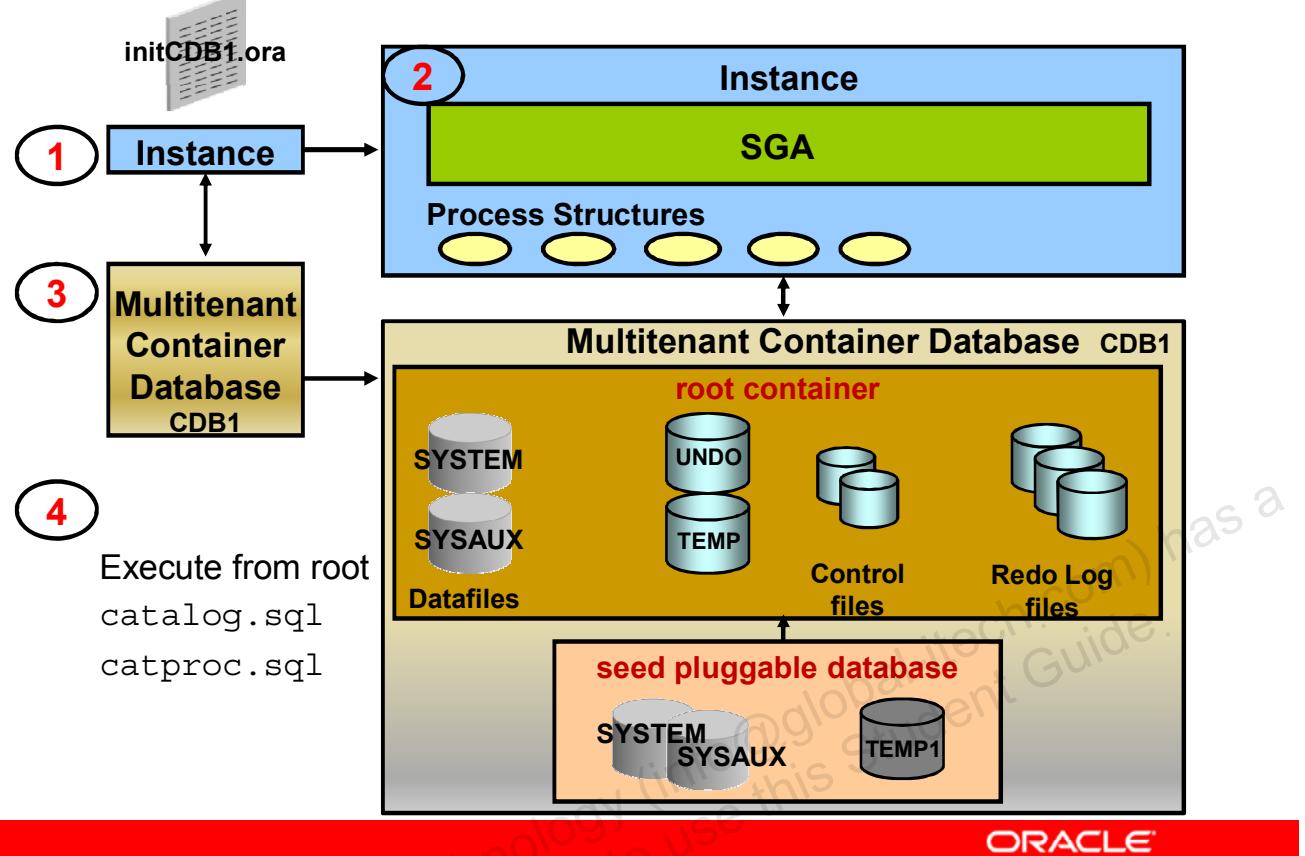
There are different tools to create and upgrade multitenant container databases. As shown in the slide, you can create a new CDB or new PDBs using either SQL*Plus or Database Configuration Assistant (DBCA) or during the installation of Oracle Database 12c. SQL Developer and Enterprise Manager Cloud Control allow you to create pluggable databases.

After they are created, use views to explore the instance, multitenant container database components, files, and pluggable databases of the CDB. Query views directly with SELECT statements using SQL*Plus, or indirectly using GUI tools such as Enterprise Manager or SQL Developer.

You can upgrade a 12.1 CDB to a 12.x CDB with Enterprise Manager or Database Upgrade Assistant (DBUA).

Note: Enterprise Manager Database Express does not provide the capability to create a CDB nor a PDB nor explore PDBs architecture.

Steps to Create a Multitenant Container Database



The steps required to create a new CDB, using either DBCA or SQL*Plus are the same.

- The first step, as for any database, non-CDB or CDB, consists of configuring an instance with an `init.ora` parameter file.
- The second step consists of starting the instance.
- The third step is the creation of the CDB using the `CREATE DATABASE` command with a new clause `ENABLE PLUGGABLE DATABASE` specifying that the database is a multitenant container database and not a non-CDB. The operation creates the root container with the control files during the mount phase, the redo log files and root data files during the open phase. The root data files are used for the **SYSTEM** tablespace containing the Oracle-supplied metadata and data dictionary, and the **SYSAUX** tablespace for AWR. It also creates the seed pluggable database with its own data files used for the **SYSAUX** and **SYSTEM** tablespaces. You may use the new clause `SEED FILE_NAME_CONVERT` to rename the data files of the seed pluggable database while copying from the root container. The clause creates the seed pluggable database and its own data files. The seed data files are copied from the root data files to another location. The seed data files can be used as templates for future PDBs creation. If you omit this clause, Oracle Managed Files determines the names and locations of the seed's files.
- The fourth step is the creation of the catalog with the execution of the `catalog.sql` and `catproc.sql` scripts connected to the root container.

Creating a Multitenant Container Database: Using SQL*Plus

1. Instance startup:

- Set ORACLE_SID=CDB1
- Set in initCDB1.ora:
 - Set CONTROL_FILES to CDB control file names
 - Set DB_NAME to CDB name
 - Set ENABLE_PLUGGABLE_DATABASE to TRUE

```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT
```

2. Create the database:

```
SQL> CREATE DATABASE CDB1 ENABLE PLUGGABLE DATABASE ...
2 SEED FILE_NAME_CONVERT ('/oracle/dbs','/oracle/seed');
```

- CDB\$ROOT container
- PDB\$SEED pluggable database

3. Close/open the seed PDB and run post-creation scripts.



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

The detailed steps to create a new CDB using SQL*Plus are the following:

- Before starting the instance, prepare an `init<SID>.ora` parameter file with the usual parameters: `DB_NAME`, `CONTROL_FILES` if OMF is not used, `DB_BLOCK_SIZE`. The global database name of the root is the global database name of the CDB. A new parameter is required to define that the instance started is ready for a CDB and not a non-CDB creation. The `ENABLE_PLUGGABLE_DATABASE` parameter must be set to `TRUE`. Set the `ORACLE_SID` environment variable. Launch SQL*Plus, connect as an OS authenticated user belonging to the DBA OS group, and execute the `STARTUP NOMOUNT` command.
- Create the CDB using the `CREATE DATABASE` command with the new clause `ENABLE PLUGGABLE DATABASE`. The clause specifies that the database is a CDB and not a non-CDB. This creates the root container and the seed pluggable database. You may use another clause `SEED FILE_NAME_CONVERT` to specify the location of the seed's files. If you omit the clause, OMF determines the names and locations of the seed's files. The `FILE_NAME_CONVERT` specifies the source directory of the root data files copied to target seed directory. Omit the clause `SEED FILE_NAME_CONVERT` if you use the new `init.ora` parameter `PDB_FILE_NAME_CONVERT`, mapping names of the root datafiles to the seed datafiles. In the example, the `/oracle/dbs` and `/oracle/seed` directories must exist. The character set defined in the statement is still the single one for the CDB.

3. Run post-creation scripts:

a. Set the session with a new parameter:

```
alter session set "_oracle_script"=true;
```

b. Close and open the seed PDB:

```
alter pluggable database pdb$seed close;
```

```
alter pluggable database pdb$seed open;
```

c. Execute catalog.sql and other post-creation scripts.

```
?/rdbms/admin/catalog.sql
```

```
?/rdbms/admin/catblock.sql
```

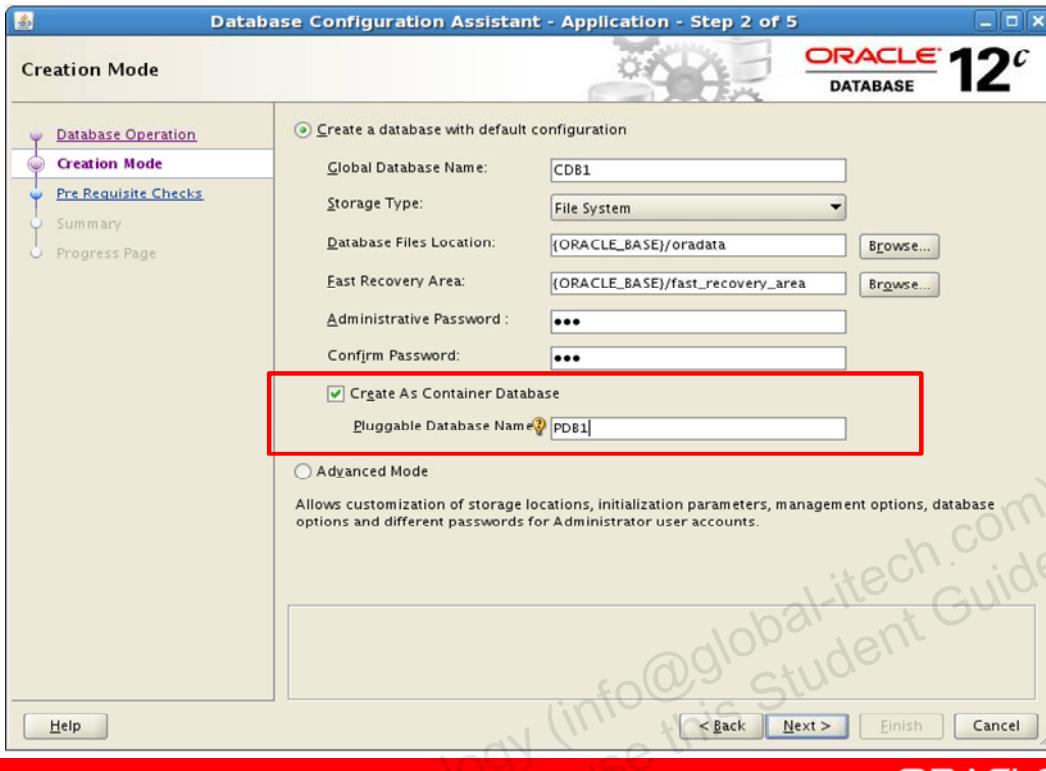
```
?/rdbms/admin/catproc.sql
```

```
?/rdbms/admin/catoctk.sql
```

```
?/rdbms/admin/owminst.plb
```

```
?/sqlplus/admin/pupbld.sql
```

Creating a Multitenant Container Database: Using DBCA



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

ORACLE

Creating the multitenant container database with DBCA requires to select the “Create As Container Database” check box, else the database is created as a non-CDB.

You also have to provide a pluggable database name when using the “Create a Database with Default Configuration” check box. If you use the “Advanced Mode” check box, you can create an empty multitenant container database with only the root and seed containers. A single

In Advanced Mode, you can register the CDB with Enterprise Manager Cloud Control and set passwords for SYS and SYSTEM users.

New Clause: SEED FILE_NAME_CONVERT

CREATE DATABASE new clauses:

```
SQL> CREATE DATABASE cdb1
  2  USER SYS IDENTIFIED BY p1 USER SYSTEM IDENTIFIED BY p2
  3  LOGFILE GROUP 1 ('/u01/app/oradata/CDB1/redo1a.log',
  4                      '/u02/app/oradata/CDB1/redo1b.log') SIZE 100M,
  5          GROUP 2 ('/u01/app/oradata/CDB1/redo2a.log',
  6                      '/u02/app/oradata/CDB1/redo2b.log') SIZE 100M
  7  CHARACTER SET AL32UTF8 NATIONAL CHARACTER SET AL16UTF16
  8  EXTENT MANAGEMENT LOCAL DATAFILE
  9          '/u01/app/oradata/CDB1/system01.dbf' SIZE 325M
10  SYSAUX DATAFILE  '/u01/app/oradata/CDB1/sysaux01.dbf' SIZE 325M
11  DEFAULT TEMPORARY TABLESPACE tempst1
12          TEMPFILE '/u01/app/oradata/CDB1/temp01.dbf' SIZE 20M
13  UNDO TABLESPACE undotbs
14          DATAFILE '/u01/app/oradata/CDB1/undotbs01.dbf' SIZE 200M
15  ENABLE PLUGGABLE DATABASE
16  SEED   FILE_NAME_CONVERT =
17          ('/u01/app/oradata/CDB1',
18              '/u01/app/oradata/CDB1/seed'),
```



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

You can find in the slide an example of a full CREATE DATABASE statement.

What is new compared to the traditional non-CDB CREATE DATABASE statement?

The first important clause required if you want the database to be a container database is ENABLE PLUGGABLE DATABASE, and one way to declare the directory for the seed datafiles is to use the SEED FILE_NAME_CONVERT clause. The FILE_NAME_CONVERT specifies the source directory of the root datafiles copied to target seed directory.

The root /u01/app/oradata/CDB1 directory and seed /u01/app/oradata/CDB1/seed directory must exist.

New Clause: ENABLE PLUGGABLE DATABASE

Without **SEED FILE_NAME_CONVERT**:

- OMF: **DB_CREATE_FILE_DEST** = '/u01/app/oradata'
- Or new instance parameter:
PDB_FILE_NAME_CONVERT =
' /u01/app/oradata/CDB1', '/u01/app/oradata/seed'

```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT

SQL> CREATE DATABASE cdb2
  2  USER SYS IDENTIFIED BY p1 USER SYSTEM IDENTIFIED BY p2
  3  EXTENT MANAGEMENT LOCAL
  4  DEFAULT TEMPORARY TABLESPACE temp
  5  UNDO TABLESPACE undotbs
  6  DEFAULT TABLESPACE users
  7  ENABLE PLUGGABLE DATABASE;
```



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

Oracle Managed Files

If you do not use explicit data file names, use Oracle Managed Files (OMF):

- Set **DB_CREATE_FILE_DEST** instance parameter with the value of the destination directory of the datafiles of the **SYSTEM**, **SYSAUX**, **UNDO**, and **USERS** tablespaces specified in the statement. Oracle chooses default sizes and properties for all datafiles, control files, and redo log files. In the example, the **/u01/app/oradata** directory must exist.

PDB_FILE_NAME_CONVERT Instance Parameter

If you do not use the **SEED FILE_NAME_CONVERT** clause, use a new instance parameter:

- **PDB_FILE_NAME_CONVERT** instance parameter maps names of existing files (the root datafiles in our case) to new file names (the seed data files in this case). In the example, both **/u01/app/oradata/CDB1** and **/u01/app/oradata/seed** directories must exist.

After CDB Creation: What's New in CDB

A CDB has new characteristics compared to non-CDBs:

- **Two containers:**
 - The **root** (CDB\$ROOT)
 - The **seed** PDB (PDB\$SEED)
- **Several services:** one per container
 - Name of root service = name of the CDB (cdb1)
- **Common** users in root and seed: SYS, SYSTEM ...
- **Common** privileges granted to common users
- **Pre-defined** common roles
- **Tablespaces** and data files associated to each container:
 - root:
 - SYSTEM: system-supplied metadata and no user data
 - SYSAUX
 - seed: SYSTEM, SYSAUX



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

When the CDB is created, there are new CDB components and objects such as:

- Two containers: the root container and the seed PDB
- As many services as containers: the service name for the root container is the CDB name given at the CDB creation concatenated with the domain name. Each new PDB is assigned a service name: the service name is the PDB name given at PDB creation concatenated with domain name. If you create or plug a PDBtest PDB, its service name would be PDBtest concatenated with domain name. You can find all service names maintained in a CDB in CDB_SERVICES view. To connect to the CDB, you connect to the root, using either local OS authentication or the root service name. For example, if you set the ORACLE_SID to the CDB instance name and use CONNECT / AS SYSDBA, you are connected to the root under common SYS user granted system privileges to manage and maintain all PDBs. To connect to a desired PDB, use either easyconnect or tnsnames.ora file.
For example, CONNECT username/password@net_service_name.
- Common users: SYS, SYSTEM, created in all containers, the root and the seed
- Common privileges granted to all users in all containers
- Pre-defined common roles in all containers, the root and the seed
- Tablespaces: SYSTEM, SYSAUX associated to each container

Data Dictionary Views: DBA_xxx

DBA_xxx All of the objects in the root or a pluggable database

ALL_xxx Objects accessible by the current user in a PDB

USER_xxx Objects owned by the current user in a PDB

DBA dictionary views providing information within PDB:

```
SQL> SELECT table_name FROM dict
  2 WHERE table_name like 'DBA%';
```

- DBA_tablespaces: All tablespaces of the PDB
- DBA_data_files: All data files of the PDB
- DBA_tables: All tables in the PDB
- DBA_users: All common and local users of the PDB



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

For backward-compatibility, DBA views show the same results in a PDB as in a non-CDB. For example the DBA_OBJECTS view shows the objects that exist in the PDB from which you run the query.

- In the root, DBA_xxx views show only objects contained in the root.
- In a PDB, the DBA_xxx views show only objects contained in the PDB.

Examples of DBA views:

- While connected to the root, you query DBA_USERS. You get the list of common users created from the root (in the root, only common users exist).
- While connected to a PDB, you query DBA_USERS. You get the list of users, common and local, of the PDB.

Data Dictionary Views: CDB_XXX

CDB_XXX

All objects in a CDB (new column CON_ID)

DBA_XXX All of the objects in the root or a pluggable database

ALL_XXX Objects accessible by the current user in a PDB

USER_XXX Objects owned by the current user in a PDB

CDB dictionary views provide information across PDBs:

```
SQL> SELECT view_name FROM dba_views
  2 WHERE view_name like 'CDB%';
```

- CDB_pdbs: All PDBs within the CDB
- CDB_tablespaces: All tablespaces within the CDB
- CDB_data_files: All datafiles within the CDB
- CDB_users: All users within the CDB (common and local)

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

In a CDB, for every DBA_* view, a CDB_* view is defined.

- In the root, CDB views can be used to obtain information about tables, tablespaces, users, privileges, parameters, PDBs and other types of objects contained in the root and all PDBs.
- In a PDB, the CDB_* views show objects visible through a corresponding DBA_* view only.

In addition, to all the columns found in a given DBA_* view, the corresponding CDB_* view also contains the CON_ID column, which identifies a container whose data a given CDB_* row represents. In a non-CDB, the value of a CON_ID column is 0. In a CDB, the value can be either 1 used for rows containing data pertaining to the root only or n where n is the applicable container ID.

Examples of CDB views:

- Connected to the root and querying CDB_USERS, you get the list of users, common and local, of each container.
- Connected to a PDB and querying CDB_USERS or DBA_USERS, you get the same list of users, common and local, of the PDB.
- Connected to the root and querying the CDB_PDBS view, you get the list of all PDBs. Querying the CDB_TABLESPACES view, you get the list of all tablespaces of all PDBs.

Data Dictionary Views: Examples

- Comparisons:

1

```
SQL> CONNECT / AS SYSDBA
SQL> SELECT role, common, con_id FROM cdb_roles;
```

2

```
SQL> SELECT role, common FROM dba_roles;
```

3

```
SQL> CONNECT sys@PDB1 AS SYSDBA
SQL> SELECT role, common, con_id FROM cdb_roles;
```

4

```
SQL> SELECT role, common FROM dba_roles;
```

- Access to data in V\$ or GV\$ views showing data from multiple PDBs can be secured using privilege.

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

NAME	OPEN_MODE
PDB\$SEED	READ ONLY
PDB1	READ WRITE
PDB2	READ WRITE

ORACLE

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

In the slide, there are some examples to make comparisons between DBA_xxx and CDB_xxx views.

- In the first example, connected to the root and querying CDB_ROLES, you get the list of roles, common and local, of each container. Note that the new column CON_ID displays the container the role belongs to.
- In the second example, querying DBA_ROLES, you get all common roles of the root only (there cannot be any local roles in the root).
- In the third example, connected to the PDB1 and querying CDB_ROLES, you get the list of roles, common and local, of the container you are connected to. The CON_ID displays the same value in all rows.
- In the forth example, querying DBA_ROLES, you get the same list except that there is no CON_ID column (useless in such queries).

The same backward-compatibility principle implies also to each of the familiar v\$views.

Data Dictionary Views: V\$xxx Views

SGA accessed by all containers: V\$ views and CON_ID column

```
SQL> SELECT distinct status, con_id FROM v$bh order by 2;
```

STATUS	CON_ID
cr	1
free	1
xcur	1
xcur	2
cr	3
xcur	3

```
SQL> select OBJECT_ID, ORACLE_USERNAME, LOCKED_MODE, CON_ID
  2  from V$LOCKED_OBJECT;
```

OBJECT_ID	ORACLE_USERNAME	LOCKED_MODE	CON_ID
83711	SYS		3
83710	DOM		3

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

In the slide, there are some examples of V\$xxx views. The new column CON_ID in V\$xxx views displays how the single SGA is accessed by any PDB within the CDB.

In the first example, the v\$bh view provides the list of distinct status of the block buffers currently in the buffer cache. The blocks are clearly identified in CON_ID column, each block being accessed by a specific container. 1 stands for the root container, 2 for the seed container and 3 by the PDB1 pluggable database.

In the second example, the v\$locked_object view provides the list of locks currently held on objects in different PDBs. The locks are clearly identified in CON_ID column as locked by a specific container; three stands for one PDB and four for another PDB.

After CDB Creation: To-Do List

After CDB creation, the CDBA has to:

- Set a separate default tablespace for the root and for each PDB
- Set a default temporary tablespace for the entire CDB (optionally create additional temporary tablespaces in individual PDBs)
- Start the listener
- Plug non-CDBs
- Test startup/shutdown procedures
- Create new event triggers to automate PDBs opening
- Create backup and recovery procedures

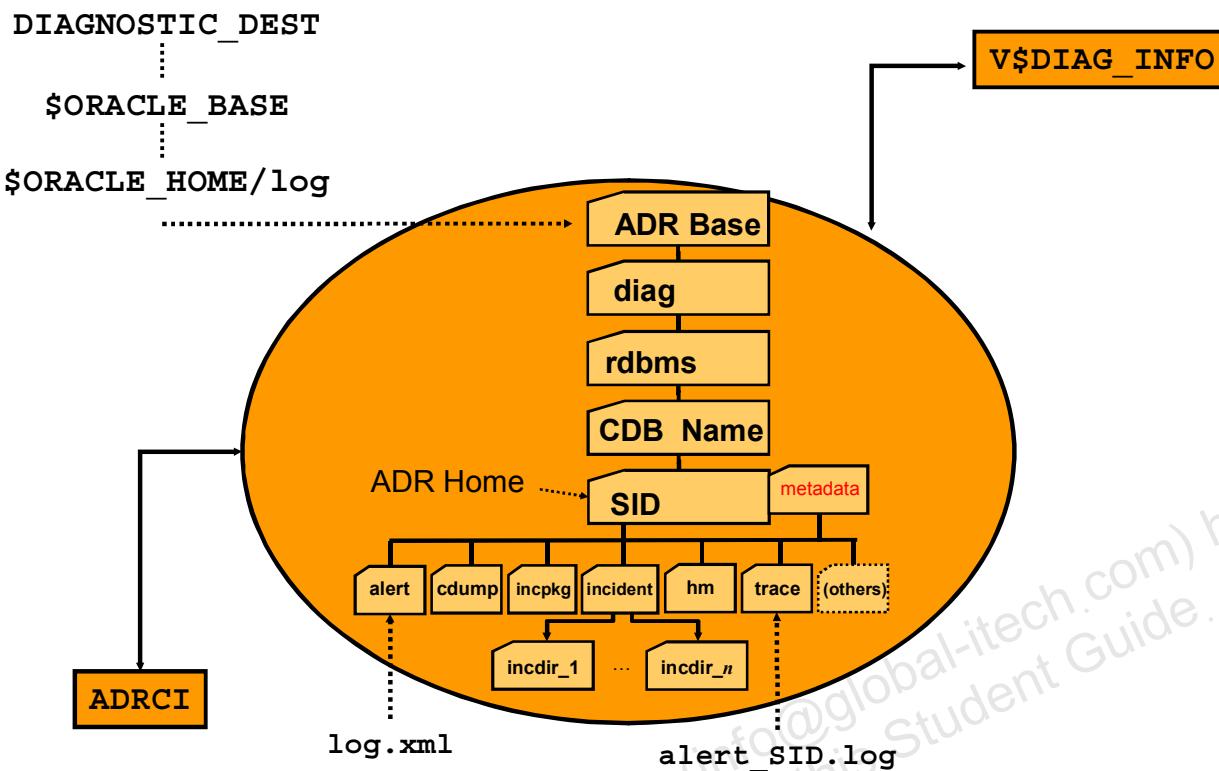


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

After the CDB is created, the multitenant container database administrator (CDBA) has to complete the following administrative tasks:

- Set a separate default tablespace for the root and for each PDB. In the root, there should not be any user data.
- Set a default temporary tablespace for the entire CDB and creating additional temporary tablespaces in individual PDBs if specific amount of temporary space is required in PDBs (this task can be performed by the pluggable database administrators, named PDBAs).
- Start the listener.
- Plug non-CDBs if your initial plan was to consolidate several non-CDBs into a single one.
- Test startup and shutdown procedures.
- Create new event triggers to automate PDBs opening.
- Create backup and recovery procedures.

Automatic Diagnostic Repository



ORACLE

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

Since Oracle Database 11g, Release 1, all traces, incident dumps and packages, the alert log, Health Monitor reports, core dumps, and more files are stored in the ADR, a file-based repository for database diagnostic data. 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 instance is down. Nothing is changed with the arrival of multitenant container databases. Each instance of each product stores diagnostic data underneath its own ADR home directory. Each CDB, linked to a single instance, stores trace files in the same ADR home directory.

Its location is set by the `DIAGNOSTIC_DEST` initialization parameter. If this parameter is omitted or left null, the database sets `DIAGNOSTIC_DEST` upon startup as follows: if the environment variable `ORACLE_BASE` is set, `DIAGNOSTIC_DEST` is set to `$ORACLE_BASE`. If the environment variable `ORACLE_BASE` is not set, `DIAGNOSTIC_DEST` is set to `$ORACLE_HOME/log`.

Automatic Diagnostic Repository: alert.log File

The alert_CDB1.log shows new DDL statements.

```
CREATE DATABASE cdb1
...
ENABLE PLUGGABLE DATABASE
SEED
FILE_NAME_CONVERT=('/u01/app/oradata/CDB1','/u01/app/oradata
/seed');

CREATE PLUGGABLE DATABASE pdb1 ... ;
ALTER PLUGGABLE DATABASE pdb1 UNPLUG INTO ... ;
ALTER PLUGGABLE DATABASE ALL OPEN ;
ALTER PLUGGABLE DATABASE pdb2 CLOSE IMMEDIATE ;
```



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

The alert.log file as a CDB shows new DDL statements such as:

- CREATE PLUGGABLE DATABASE
- ALTER PLUGGABLE DATABASE
- DROP PLUGGABLE DATABASE

Quiz

Which is a characteristic of the seed pluggable database of a CDB?

- a. It is always kept in READ ONLY mode.
- b. It is not a container.
- c. The seed can be dropped.

Quiz

You create a CDB. What is true about the seed pluggable database?

- a. Copy the seed data files yourself.
- b. Use the new clause `SEED FILE_NAME_CONVERT` in the `CREATE DATABASE` statement.
- c. The seed pluggable database is not required.
- d. The seed pluggable database does not require data files.



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

Answer: b

Practice 3 Overview: Creating a CDB and PDBs

The first practice covers the following topic:

- Creating a CDB with no PDBs

Provisioning New Pluggable Databases

Four methods:

- Create a new PDB from the seed PDB.
- Plug a non-CDB in a CDB.
- Clone a PDB from another PDB:
 - Into the same CDB
- Plug an unplugged PDB into another CDB.



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

There are four methods to provision new PDBs in a CDB.

- Create a new PDB from the seed, the PDB\$SEED PDB, for example for a new application implementation. This type of PDB creation is nearly instantaneous.
- Plug non-CDBs in a CDB as PDBs, as part of migration strategy. It is also a good way to consolidate several non-CDBs into a CDB.
- Clone a PDB from another PDB of the same CDB. For example, you want to test the application patch of your production. You first clone your production application in a cloned PDB and patch the cloned PDB to test.
- Plug an unplugged PDB into another CDB. For example, you have to upgrade a PDB to the latest Oracle version, but you do not want to apply it on all PDBs. Instead of upgrading a CDB from one release to another, you can unplug a PDB from one Oracle Database release, and then plug it into a newly created CDB from a higher release.

Tools

To provision new PDBs, you can use:

- SQL*Plus
- SQL Developer
- Enterprise Manager Cloud Control
- DBCA
 - Copy from seed
 - By unplugging / plugging method



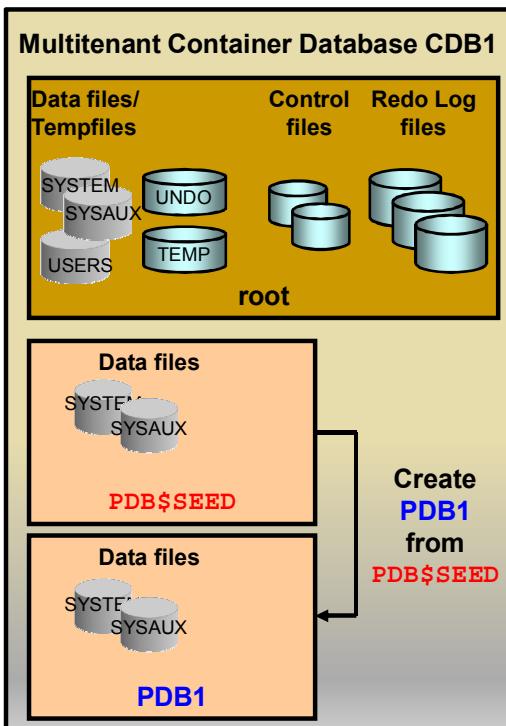
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

There are different tools to provision new PDBs in a CDB.

- SQL*Plus
- SQL Developer
- Enterprise Manager Cloud Control

To create a new PDB from seed or by unplug/plug method, you can also use DBCA.

Method 1: Create New PDB from PDB\$SEED



- Copies the data files from PDB\$SEED data files
- Creates tablespaces SYSTEM, SYSAUX
- Creates a full catalog including metadata pointing to Oracle-supplied objects
- Creates common users:
 - Superuser SYS
 - SYSTEM
- Creates a local user (PDBA) granted local PDB_DBA role
- Creates a new default service

ORACLE

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

The creation of a new PDB from the seed is nearly instantaneous. The operation copies the data files from the READ ONLY seed PDB to the target directory defined in the CREATE PLUGGABLE DATABASE statement.

It creates tablespaces such as SYSTEM to store a full catalog including metadata pointing to Oracle-supplied objects, SYSAUX for local auxiliary data.

It creates default schemas and common users that exist in seed PDB, SYS who continues to have all superuser privileges and SYSTEM who can administer the PDB.

It creates a local user (the PDBA) granted a local PDB_DBA role. Until the PDB SYS user grants privileges to the local PDB_DBA role, the new PDBA cannot perform any other operation than connecting to the PDB.

A new default service is also created for the PDB.

Steps: With FILE_NAME_CONVERT

Create a new PDB from the seed using **FILE_NAME_CONVERT**:

1. Connect to the root as a common user with CREATE PLUGGABLE DATABASE system privilege:

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  2  ADMIN USER admin1 IDENTIFIED BY p1 ROLES=(CONNECT)
  3  FILE_NAME_CONVERT = ('PDB$SEEDdir', 'PDB1dir');
```

2. Use views to verify:

```
SQL> CONNECT / AS SYSDBA
SQL> SELECT * FROM cdb_pdbs;
SQL> SELECT * FROM cdb_tablespaces;
SQL> SELECT * FROM cdb_data_files;
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN RESTRICTED;
SQL> CONNECT sys@pdb1 AS SYSDBA
SQL> CONNECT admin1@pdb1
```

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

The steps to create a new PDB from the seed are the following:

If you do not use OMF (Oracle Managed Files):

1. Connect to the root as a common user with CREATE PLUGGABLE DATABASE system privilege and execute the CREATE PLUGGABLE DATABASE statement as shown in the slide. The ADMIN_USER clause defines the PDBA user created in the new PDB with the CONNECT and PDB_DBA roles (empty role). The clause FILE_NAME_CONVERT designates first the source directory of the seed data files and second the destination directory for the new PDB data files.
2. When the statement completes, use views to verify that the PDB is correctly created.
 - The CDB_PDBS view displays the list of the PDBs, the CDB_TABLESPACES view displays the list of the tablespaces of the new PDB (SYSTEM, SYSAUX).
 - Still connected to the root, open the PDB. Then try to connect to the new PDB under common user SYS who always exists in any PDB or admin1.

Steps: Without FILE_NAME_CONVERT

Create a new PDB from seed without **FILE_NAME_CONVERT**:

- OMF: **DB_CREATE_FILE_DEST** =
 '/u01/app/oradata/CDB1/**pdb1**'

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  2  ADMIN USER pdb1_admin IDENTIFIED BY p1
  3  ROLES=(CONNECT);
```

Or

- New parameter: **PDB_FILE_NAME_CONVERT** =
 '/u01/app/oradata/CDB1/seed','/u01/app/oradata/CDB1/**pdb1**'

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  2  ADMIN USER pdb1_admin IDENTIFIED BY p1
  3  ROLES=(CONNECT);
```

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

If you use OMF or PDB_FILE_NAME_CONVERT, then first connect to the root of the CDB as a SYS.

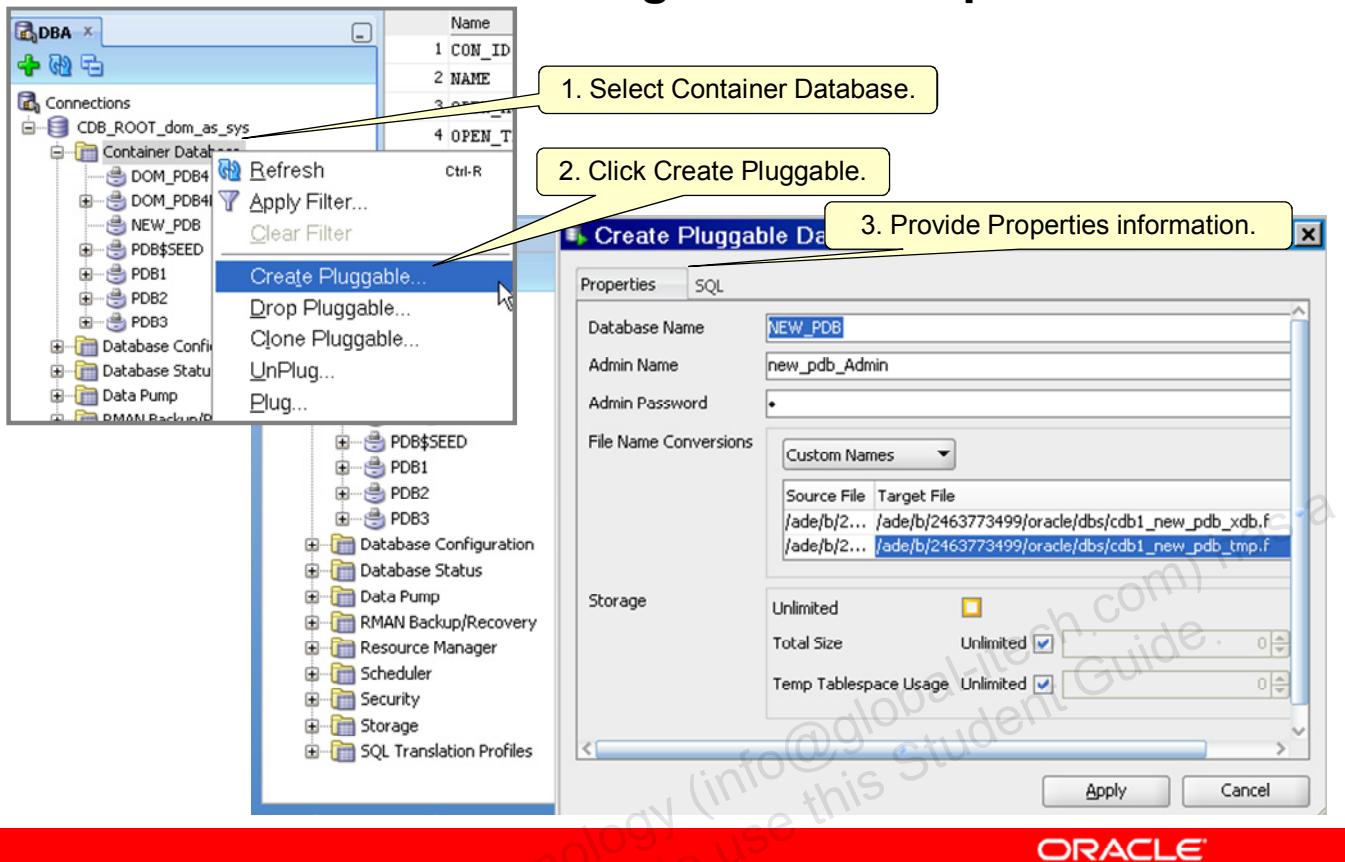
- With OMF, set in `init.ora` `DB_CREATE_FILE_DEST` instance parameter to a target directory for the data files of the new PDB.
- Without OMF, set `PDB_FILE_NAME_CONVERT` new instance parameter to both the source directory of the seed datafiles and the target directory for the new PDB data files.

Both `/u01/app/oradata/CDB1/pdb1` and `/u01/app/oradata/CDB1/pdb2` directories must exist.

Then use `cdb_pdbs` view to verify that the new PDB and its tablespaces exist:

```
SQL> SELECT * FROM cdb_pdbs;
SQL> SELECT * FROM cdb_tablespaces;
SQL> SELECT * FROM cdb_data_files;
```

Method 1: Using SQL Developer



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

You can use SQL*Developer to perform the same type of operations, such as create a new PDB from the seed, clone a PDB from another PDB from the CDB or from another CDB.

First of all, click to view the CDB and its PDBs, click View option from the top menu then DBA.

If you want to create a new PDB from the seed, then select Container Database, then choose the Create Pluggable option, then provide the PDB name, the PDB administrator user name 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 seed data files used as templates during the copy.

Method 1: Using SQL Developer

4. View SQL statement.

```
CREATE PLUGGABLE DATABASE NEW_PDB ADMIN USER new_pdb_Admin IDENTIFIED BY p
FILE_NAME_CONVERT=(
  '/ade/b/2463773499/oracle/dbs/cdbl_pdb0_db.f', '/ade/b/2463773499/oracle/dbs/cdbl_new_pdb_db.f',
  '/ade/b/2463773499/oracle/dbs/cdbl_pdb0_ax.f', '/ade/b/2463773499/oracle/dbs/cdbl_new_pdb_ax.f',
  '/ade/b/2463773499/oracle/dbs/cdbl_pdb0_xdb.f', '/ade/b/2463773499/oracle/dbs/cdbl_new_pdb_xdb.f',
  '/ade/b/2463773499/oracle/dbs/cdbl_pdb0_tmp.f', '/ade/b/2463773499/oracle/dbs/cdbl_new_pdb_tmp.f'
)
STORAGE UNLIMITED
```

5. After PDB is created:

- CON_ID defines the new container identifier in the CDB.
- OPEN_MODE defines the open status, by default MOUNTED.
- GUID defines the new global unique container identifier.

Name	Value
1 CON_ID	8
2 NAME	NEW_PDB
3 OPEN_MODE	MOUNTED
4 OPEN_TIME	05-JAN-12 09.03.39.600000000 AM
5 DBID	1221202379
6 CON_UID	84955
7 GUID	B5CBE804C06B46FBE0431C72E50A5FB4
8 CREATE_SCN	1669072
9 OPEN_TIME	05-JAN-12 09.03.39.600000000 AM

ORACLE

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

Before applying the creation, you can view the SQL statement.

After it is created, the PDB is assigned a CON_ID number, unique within the CDB and a GUID global unique identifier.

Synchronization

1. Customer-created common users or roles in root:
 - Cannot be created, modified, dropped when PDB is in READ-ONLY mode
 - Can be created, modified, dropped when PDB is in MOUNTED mode
2. When opening the PDB:
 - In READ-ONLY mode, an error is returned.
 - In READ-WRITE mode, synchronization with the target CDB is automatically completed.
 - A compatibility check is automatically performed:
 - Any violation is reported in the `PDB_PLUG_IN_VIOLATIONS` view.
 - If there are no violation, the PDB status is changed to NORMAL.



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

Once a PDB is created using seed PDB or plugging or cloning methods, or even closed, you can view the status of the new or closed PDB by querying the STATUS column of the `CDB_PDBS` view.

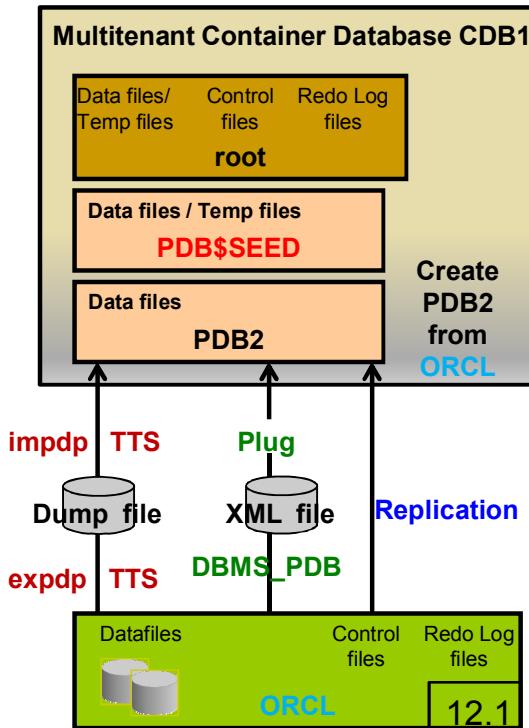
If common users and roles had been previously created, the new or closed PDB must be synchronized to retrieve the new common users and roles from the root. The synchronization is automatically performed if the PDB is opened in read write mode.

If you open the PDB in read only mode, an error is returned.

When a PDB is opened, Oracle Database checks the compatibility of the PDB with the CDB.

Each compatibility violation is either a warning or an error. If a compatibility violation is a warning, then the warning is recorded in the alert log, but the PDB is opened normally without displaying a warning message. If a compatibility violation is an error, then an error message is displayed when the PDB is opened, and the error is recorded in the alert log. You must correct the condition that caused each error. When there are errors, access to the PDB is limited to users with `RESTRICTED SESSION` privilege so that the compatibility violations can be addressed. You can view descriptions of violations by querying `PDB_PLUG_IN_VIOLATIONS` view.

Method 2: Plug a Non-CDB into CDB



Three possible methods:

- TTS or TDB or full export/import
- XML file definition with DBMS_PDB
- Replication

Entities are created in the new PDB:

- Tablespaces: SYSTEM, SYSAUX
- A full catalog
- Common users: SYS, SYSTEM
- A local administrator (PDBA)
- A new default service

ORACLE

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

There are three possible methods to plug a non-CDB database into a CDB.

Whichever method is used, you have to get the non-CDB into a transactionally-consistent state and open it in restricted mode.

- Either use transportable tablespace (TTS) or full conventional export / import or transportable database (TDB) provided that in the last one any user-defined object resides in a single user-defined tablespace.
- Or use DBMS_PDB package to construct an XML file describing the non-CDB data files to plug the non-CDB into the CDB as a PDB. This method presupposes that the non-CDB is an Oracle 12c database.
- Or use replication with GoldenGate

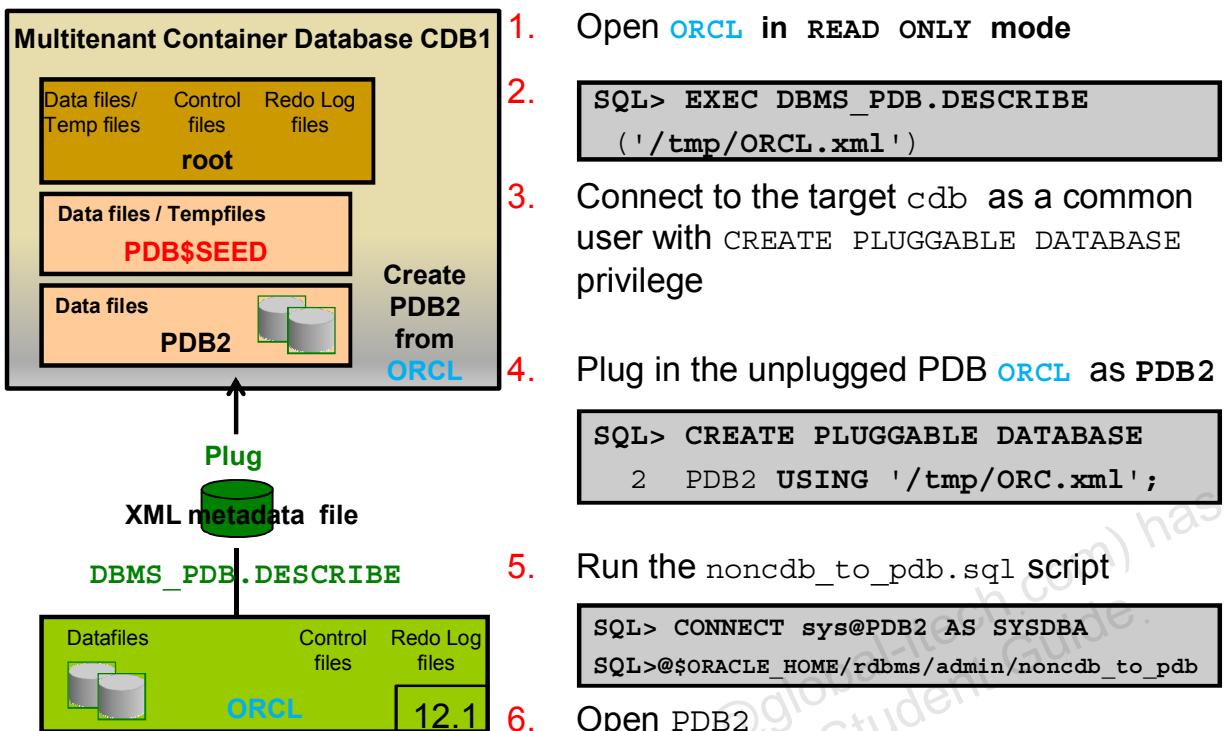
Using the DBMS_PDB package is the easiest option.

If the DBMS_PDB package is not used, then using export/import is usually simpler than using GoldenGate replication, but export/import might require more down time during the switch from the non-CDB to the PDB.

If you choose to use export/import, and you are moving a whole non-CDB into the CDB, then transportable databases (TDB) is usually the best option. If you choose to export and import part of a non-CDB into a CDB, then transportable tablespaces (TTS) is the best option.

To understand the steps of the first or third method, refer to the following appendix of the course “Appendix B – Other PDB Creation Methods”.

Plug a Non-CDB in to CDB Using DBMS_PDB



Note: The STATUS of the PDB is CONVERTING.

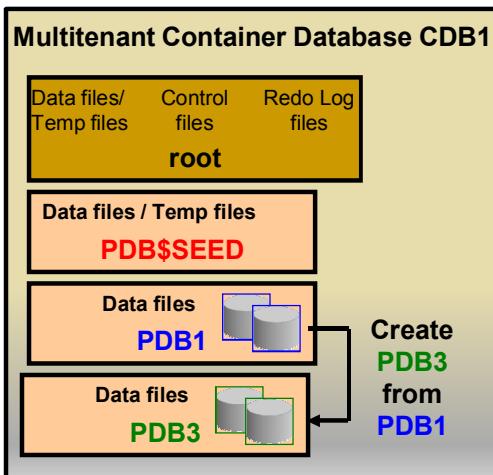
ORACLE

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

The technique with DBMS_PDB package creates an unplugged PDB from an Oracle database 12c non-CDB. The unplugged PDB can then be plugged in to a CDB as a new PDB. Running the DBMS_PDB.DESCRIBE procedure on the non-CDB generates an XML file that describes the future PDB. You can plug in the unplugged PDB in the same way that you can plug in any unplugged PDB, using the XML file and the non-CDB data files. The steps are the following:

1. Connect to non-CDB ORCL and first ensure that the non-CDB ORCL is in a transactionally-consistent state and place it in read-only mode.
2. Execute the DBMS_PDB.DESCRIBE procedure, providing the file name that will be generated. The XML file contains the list of data files to be plugged. The XML file and the data files described in the XML file comprise an unplugged PDB.
3. Connect to the target CDB to plug the unplugged ORCL as PDB2.
4. Before plugging the unplugged PDB, make sure it can be plugged into a CDB using the DBMS_PDB.CHECK_PLUG_COMPATIBILITY procedure. Execute the CREATE PLUGGABLE STATEMENT using the new clause USING 'XMLfile'. The list of data files from ORCL is read from the XML file to locate and name the data files of PDB2.
5. Run the ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql script to delete unnecessary metadata from PDB SYSTEM tablespace. This script must be run before the PDB can be opened for the first time. This script is required for plugging non-CDBs only.
6. Open PDB2 to verify that the application tables are in PDB2.

Method 3: Clone PDBs



PDB3 owns:

- SYSTEM, SYSAUX tablespaces
- Full catalog
- SYS, SYSTEM common users:
- Same local administrator name
- New service name

1. In init.ora, set DB_CREATE_FILE_DEST= 'PDB3dir' or
PDB_FILE_NAME_CONVERT='PDB1dir', 'PDB3dir'

2. Connect to the root.

3. Quiesce PDB1 (Close PDB1 before):

```
SQL> ALTER PLUGGABLE DATABASE
2 pdb1 OPEN READ ONLY;
```

4. Clone PDB3 from PDB1:

```
SQL> CREATE PLUGGABLE DATABASE
2 pdb3 FROM pdb1;
```

5. Open PDB3 in read-write mode.

ORACLE

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

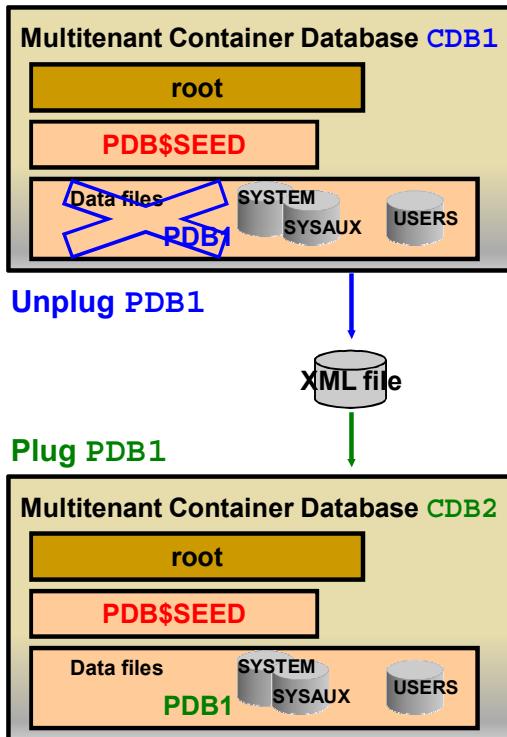
This technique copies a source PDB from a CDB and plugs the copy in to a CDB. The source PDB is in the local CDB.

The steps to clone a PDB within the same CDB are the following:

1. In init.ora, set DB_CREATE_FILE_DEST= 'PDB3dir' (OMF) or
PDB_FILE_NAME_CONVERT='PDB1dir', 'PDB3dir' (non OMF).
2. Connect to the root of the CDB as a common user with CREATE PLUGGABLE DATABASE privilege.
3. Quiesce the source PDB used to clone, using the command ALTER PLUGGABLE DATABASE pdb1 READ ONLY after closing the PDB using the command ALTER PLUGGABLE DATABASE CLOSE
4. Use the command CREATE PLUGGABLE DATABASE to clone the PDB pdb3 FROM pdb1.
5. Then open the new pdb3 with the command ALTER PLUGGABLE DATABASE OPEN.

If you do not use OMF, in step 4, use the command CREATE PLUGGABLE DATABASE with the clause FILE_NAME_CONVERT='(pdb1dir', ' pdb3dir') to define the directory of the source files to copy from PDB1 and the target directory for the new files of PDB3.

Method 4: Plug Unplugged PDB in to CDB



Unplug **PDB1** from **CDB1**:

1. Connect to **CDB1** as a common user.
2. Verify that **PDB1** is opened READ ONLY.
3.

```
SQL> ALTER PLUGGABLE DATABASE
2   pdb1 UNPLUG INTO
3       'xmlfile1.xml';
```
4. Drop **PDB1** from **CDB1**.

Plug **PDB1** in to **CDB2**:

1. Connect to **CDB2** as a common user.
2. Use DBMS_PDB package to check the compatibility of **PDB1** with **CDB2**.
3.

```
SQL> CREATE PLUGGABLE DATABASE
2   pdb1 USING 'xmlfile1.xml'
3   NOCOPY;
```
4. Open **PDB1** in read-write mode.

ORACLE

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

You can create a PDB in a CDB by the unplugging / plugging method.

Unplugging a PDB disassociates the PDB from a CDB. You unplug a PDB when you want to move the PDB to a different CDB, or when you no longer want the PDB to be available.

The first step is to unplug PDB1 from CDB1. The second step is to plug PDB1 in to CDB2.

To unplug PDB1 from CDB1, first connect to the root of CDB1, and check that the PDB is in READ ONLY mode using V\$PDBS view. Then use ALTER PLUGGABLE DATABASE with UNPLUG clause to specify the database to unplug and the XML file to unplug it into. A PDB must be dropped from the CDB before it can be plugged back in to the same CDB. If the PDB is plugged in to another CDB, the PDB does not need to be dropped if the data files are copied.

Before plugging PDB1 in to CDB2, you can optionally check whether the unplugged PDB is compatible with the CDB2 with DBMS_PDB.CHECK_PLUG_COMPATIBILITY function.

```
SET SERVEROUTPUT ON
DECLARE compat BOOLEAN;
BEGIN
  compat := DBMS_PDB.CHECK_PLUG_COMPATIBILITY( pdb_descr_file
                                             => '/disk1/usr/salespdb.xml',
                                             pdb_name => 'pdb1');
  DBMS_OUTPUT.PUT_LINE('Is PDB compatible? = ' || compat);
END;
```

To plug PDB1 in to CDB2, connect to CDB2 root as a common user and use CREATE PLUGGABLE DATABASE pdb1 USING 'xmlfile1'. The last step is opening the PDB.

Method 4: Flow

Several clauses can be used in conjunction:

Are new PDB files based on same files that were used to create existing PDB in CDB?

If not, AS CLONE clause is required and so, it ensures that Oracle Database generates unique PDB DBID, GUID, and other identifiers expected for the new PDB.

XML file accurately describes current locations of files?

If not, the SOURCE_FILE_NAME_CONVERT clause is required.

Are files are in correct location?

If not, specify COPY to copy files to new location or MOVE to move them to another location.

If yes, use NOCOPY. COPY is the default.

- FILE_NAME_CONVERT clause of CREATE PLUGGABLE DATABASE statement
- OMF: DB_CREATE_FILE_DEST parameter
- PDB_FILE_NAME_CONVERT parameter

Do you want to specify storage limits for PDB?

If yes, specify the STORAGE clause.

ORACLE

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

Several clauses can be used in conjunction according to different assumptions:

First question: Are the files of the new PDB based on the same files that were used to create an existing PDB in CDB? If not, the AS CLONE clause is required and so, it ensures that Oracle Database generates unique PDB DBID, GUID, and other identifiers expected for the new PDB.

Second question: Does the XML file accurately describe the current locations of the files? If not, the SOURCE_FILE_NAME_CONVERT clause is required.

Third question: Are the files are in the correct location? If not, specify COPY to copy the files to a new location or MOVE to move them to another location. If yes, use NOCOPY. COPY is the default.

Use one of these techniques to specify the target location:

- In the CREATE PLUGGABLE DATABASE statement, include a FILE_NAME_CONVERT clause that specifies the target locations based on the names of the source files.
- Or enable OMF to determine the target location using DB_CREATE_FILE_DEST initialization parameter.
- Or specify the target locations in the PDB_FILE_NAME_CONVERT initialization parameter.

If you use more than one of these techniques, then the precedence order is:

- FILE_NAME_CONVERT clause
- Oracle Managed Files using DB_CREATE_FILE_DEST initialization parameter
- PDB_FILE_NAME_CONVERT initialization parameter

Fourth question: Do you want to specify storage limits for the PDB? If yes, specify the STORAGE clause.

That's why according to the different combinations, you can use different syntaxes:

```
CREATE PLUGGABLE DATABASE PDB1AS CLONE USING
'/disk1/usr/salespdb.xml' COPY;

CREATE PLUGGABLE DATABASE PDB1USING '/disk1/usr/salespdb.xml'
SOURCE_FILE_NAME_CONVERT = ('/disk1/oracle/sales',
'/disk2/oracle/sales') NOCOPY STORAGE (MAXSIZE 500M
MAX_SHARED_TEMP_SIZE 100M);

CREATE PLUGGABLE DATABASE PDB1USING '/disk1/usr/salespdb.xml' COPY
FILE_NAME_CONVERT = ('/disk1/oracle/sales',
'/disk2/oracle/sales');
```

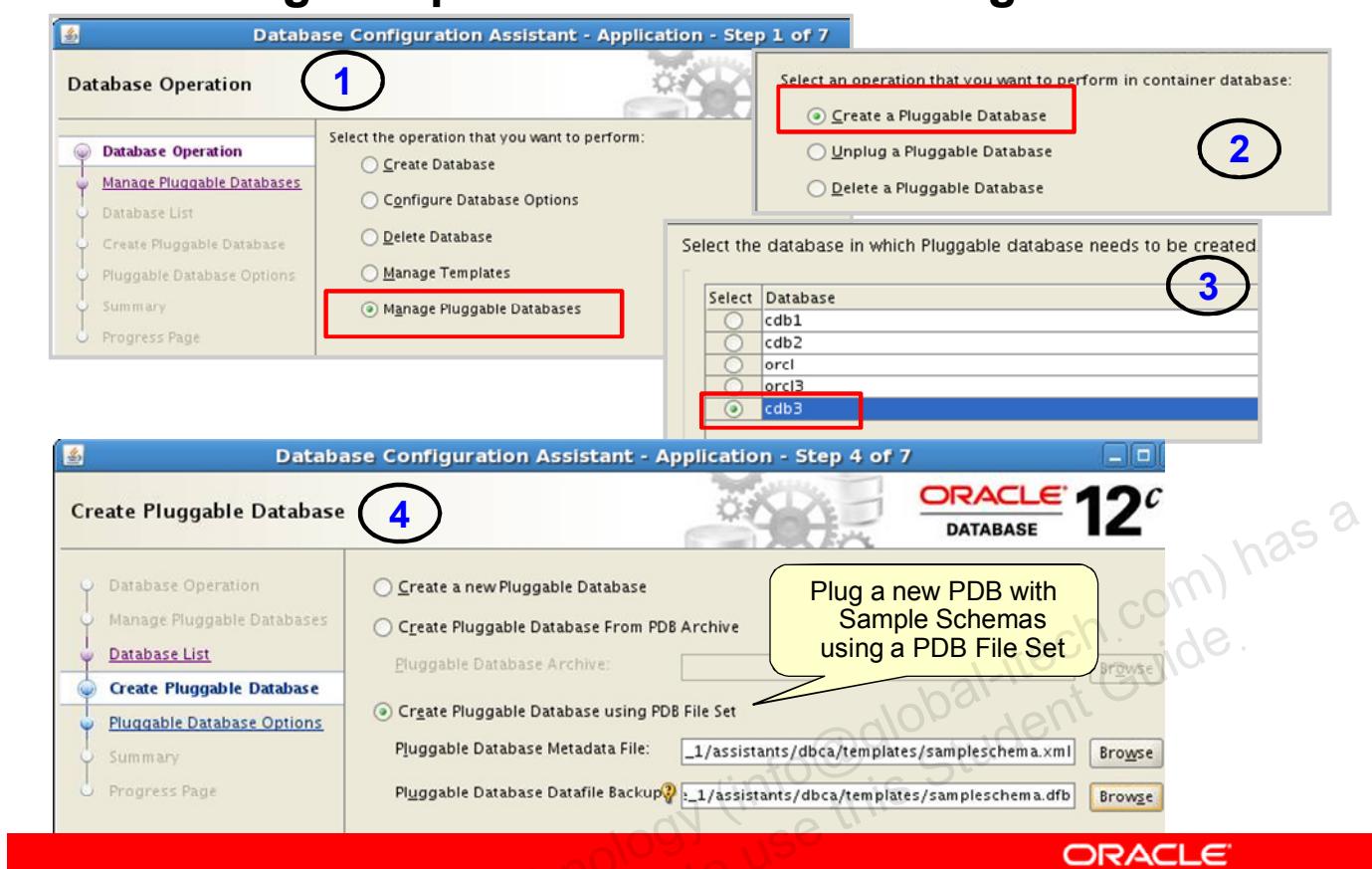
To know if a PDB is unplugged, display the STATUS column in CDB_PDBS view: the value is UNPLUGGED:

```
SQL> select pdb_name , status from CDB_PDBS;
```

PDB_NAME	STATUS
PDB1	NORMAL
PDB\$SEED	NORMAL
PDB3	UNPLUGGED

Plug Sample Schemas PDB: Using DBCA

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

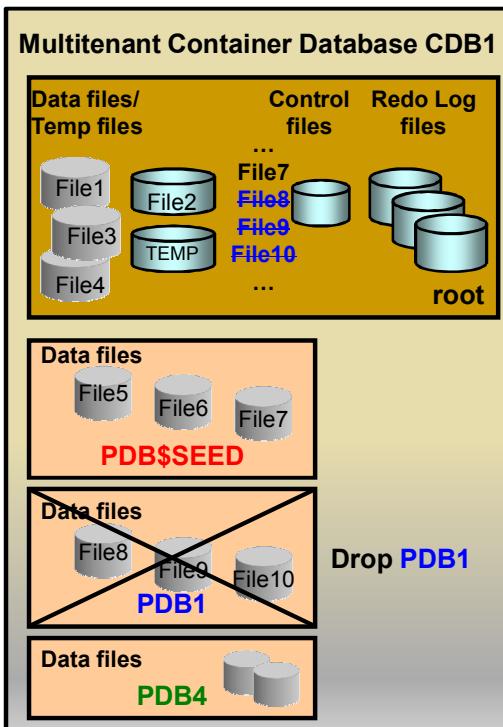


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can plug a new PDB with the sample schemas using DBCA.

1. In DBCA, click the Manage Pluggable Databases.
2. Then choose Create a Pluggable Database.
3. Select the CDB in which you intend to create the new PDB.
4. Click Create Pluggable Database using PDB File Set. Browse until you find both files:
 - Pluggable Database Metadata File:
\$ORACLE_HOME/assistants/dbca/templates/sampleschema.xml
 - Pluggable Database Datafile Backup:
\$ORACLE_HOME/assistants/dbca/templates/sampleschema.dfb
5. Define a name for the new PDB and a location for the data files.
You can also define a PDB User to create a new administrator for the PDB.
6. Click Next and Finish.

Dropping a PDB



```
SQL> ALTER PLUGGABLE DATABASE
2  pdb1 CLOSE;
SQL> DROP PLUGGABLE DATABASE
2  pdb1 [INCLUDING DATAFILES];
```

- Updates control files
- If INCLUDING DATAFILES :
 - Removes **PDB1** datafiles
- If KEEP DATAFILES (default):
 - Retain data files
 - Can be plugged in another or the same CDB
- Requires SYSDBA privilege
- Cannot drop seed PDB

ORACLE

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

When you no longer need the data in a PDB, you can drop the PDB.

You can also unplug it and then drop it specifying `KEEP DATAFILES` to retain the data files associated with the PDB after the PDB is dropped. `KEEP DATAFILES` is the default behavior.

Keeping data files may be useful in scenarios where an unplugged PDB is plugged in to another CDB.

When you drop a PDB specifying `INCLUDING DATAFILES`, all of its data files listed in the control file are deleted.

With or without the clause `INCLUDING DATAFILES`, the `DROP PLUGGABLE DATABASE` statement modifies the control files to eliminate all references to the dropped PDB.

PDB backups are not removed, but you can use RMAN to remove them. This operation requires the `SYSBACKUP` privilege.

To perform the operation, use either SQL*Plus or SQL Developer or DBCA or Enterprise Manager Cloud Control.

You cannot drop the seed PDB.

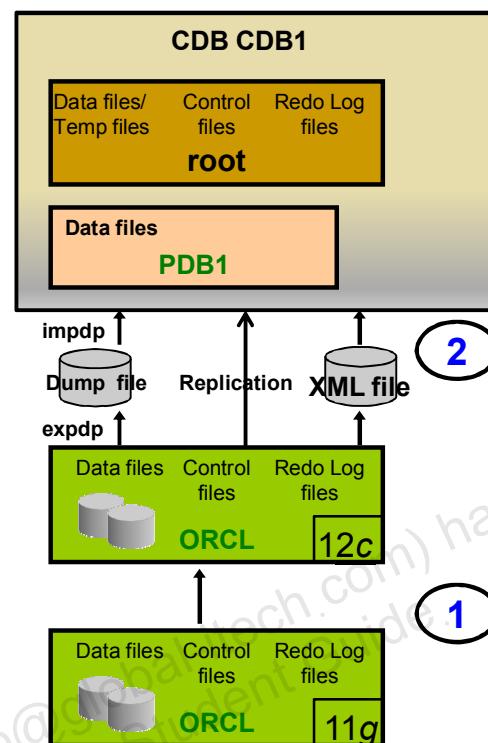
Migrating pre-12.1 Databases to 12.1 CDB

There are two methods:

1. Upgrade an existing pre-12.1 database to 12c.
2. Plug-in non-CDB into a CDB.

Or

1. Pre-create a PDB in CDB.
2. Use 11g expdp / 12c impdp or replication between non-CDB and PDB.



ORACLE

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

There are two methods to migrate a non-CDB 11g database to a 12c CDB.

The first method consists of two steps:

1. Upgrade the 11g database to 12c non-CDB.
2. Plug in the 12c non-CDB into a CDB: Use `DBMS_PDB.DESCRIBE` procedure to generate the XML file to plug the data files into the CDB as a new PDB.

The second method consists of two steps:

1. Pre-create a PDB in the CDB from the seed PDB. This operation establishes a Oracle Database 12c dictionary in the newly created PDB.
2. Use either export/import or replication to load the 11g data into the newly created PDB of the CDB.

Quiz

Which of the following are true about cloning a PDB into the same CDB? Select all that apply.

- a. It is not possible. You can only clone a PDB into another CDB.
- b. You can clone only one PDB into the same CDB.
- c. Cloning a PDB can use the source files copy method to the target PDB files.
- d. Cloning a PDB can use the clause NOCOPY if the target PDB files will use the source files.



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

Answer: c

Quiz

Which of the following are true about dropping a PDB?

- a. You can drop a PDB only if the PDB is closed.
- b. You can possibly drop the seed PDB, but you will not be able to create any other PDB within the CDB.
- c. You can drop a PDB and keep the data files to be reused by another PDB.
- d. When you drop a PDB, the data files and redo log files are automatically removed from the storage file system.



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

Answer: a, c

Summary

In this lesson, you should have learned how to:

- Configure and create a CDB
- Create a PDB from PDB\$SEED
- Create a PDB from a non-CDB
- Clone a PDB into the same CDB
- Unplug and plug a PDB from one CDB to another a CDB
- Explore the instance and structure of PDBs
- Drop a PDB
- Migrate pre-12.1 non-CDB database to CDB



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

Practice 3 Overview: Creating a CDB and PDBs

These practices cover the following topics:

- Creating a new PDB into a CDB using the seed
- Cloning a PDB from a CDB into the same CDB
- Plugging a non-CDB in to a CDB
- Merging two CDBs into a single one
- Dropping a PDB



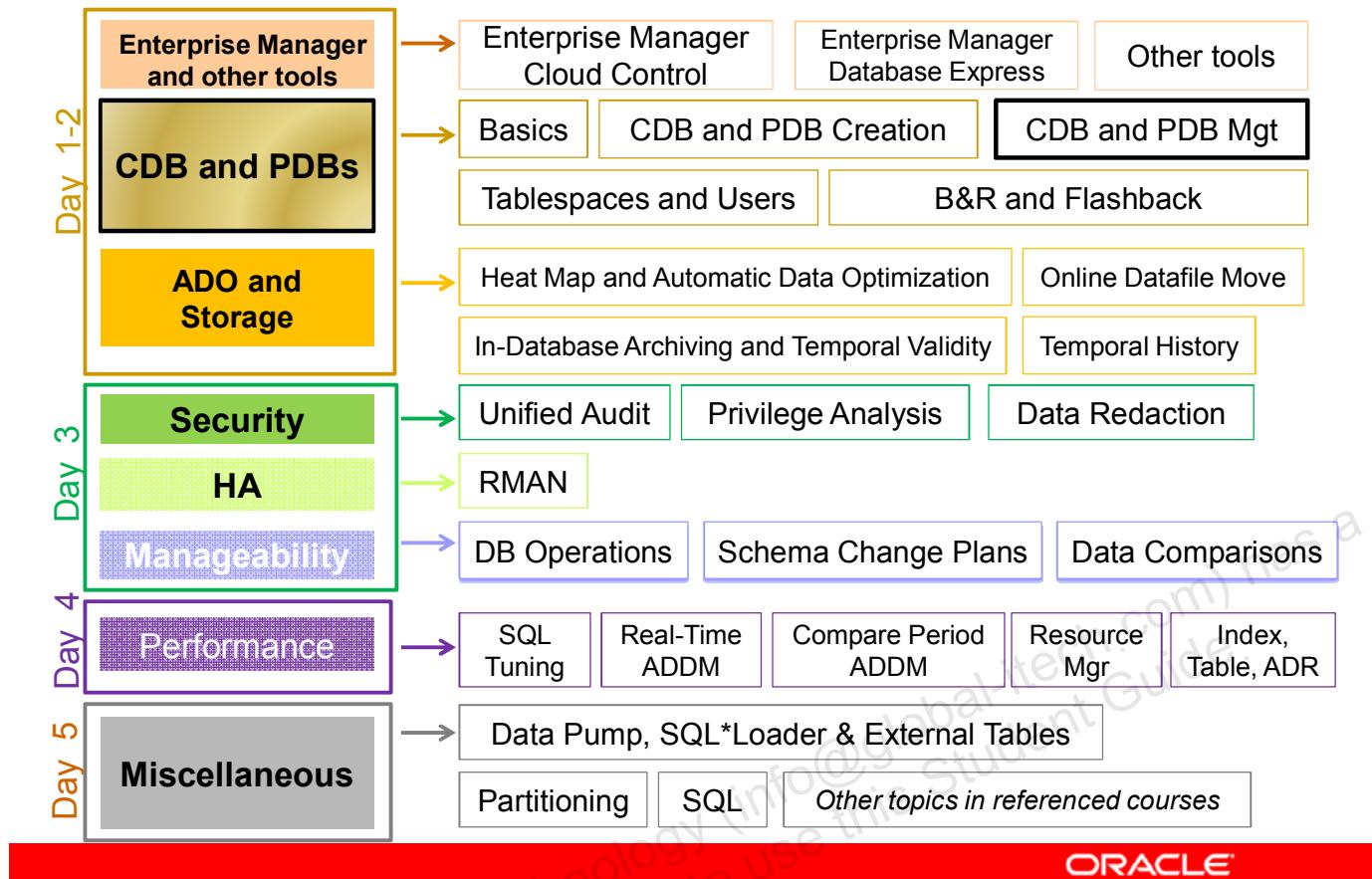
Managing Multitenant Container Databases and Pluggable Databases



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.

Objectives

After completing this lesson, you should be able to:

- Establish connections to CDB / PDB
- Start up and shut down a CDB
- Open and close PDBs
- Create event triggers to open PDBs
- Change the different modes and settings of PDBs
- Evaluate the impact of parameter value changes



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

Note

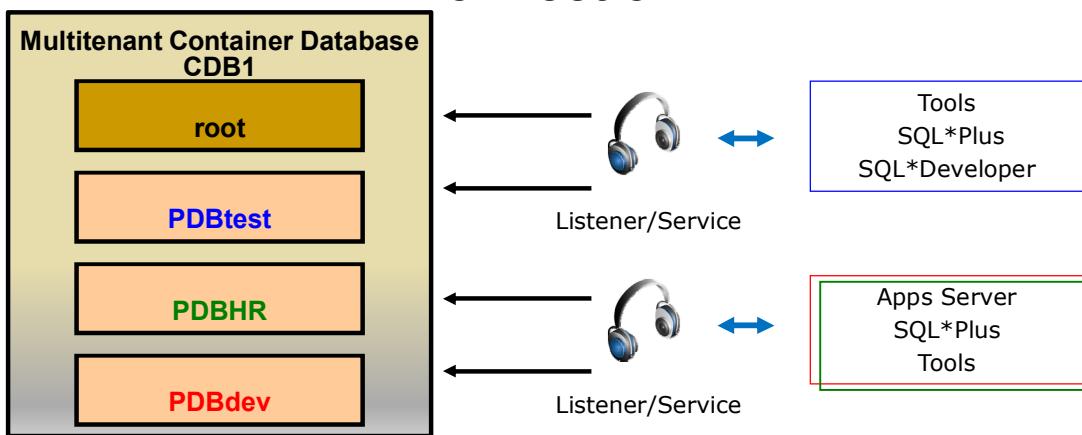
For a complete understanding of the multitenant architecture and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Administrator's Guide 12c Release 1 (12.1)*

Refer to other sources of information:

- *Oracle Database 12c New Features Demo Series* demonstrations under Oracle Learning Library:
 - *CDB and PDBs Administration – Demo 1: Open and Close CDBs and PDBs*
 - *CDB and PDBs Administration – Demo 2: Change PDB mode*

Connection



- Every PDB has a default service.

```
SQL> SELECT name, pdb FROM cdb_services;
```

- Service name has to be unique across CDBs.

```
SQL> CONNECT / AS SYSDBA
SQL> CONNECT sys@PDBtest AS SYSDBA
SQL> CONNECT local_user1@hostname1:1525/PDBHR
SQL> CONNECT common_user2@PDBdev
SQL> SHOW CON_NAME
```

ORACLE

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

Before performing any maintenance and management operation in PDBs, you examine how to connect to a CDB and to a particular PDB. Any container in a CDB owns a service name.

- The root container service name is the CDB name given at the CDB creation concatenated with domain name.
- Each new PDB is assigned a service name: the service name is the PDB name given at PDB creation concatenated with domain name. If you create or plug a PDBtest PDB, its service name is PDBtest concatenated with domain name. The container service names must be unique within a CDB, and even across CDBs that register with the same listener.
- You can find all service names maintained in a CDB and PDBs in CDB_SERVICES or V\$SERVICES views. The PDB column shows the PDB to which the services are linked.

To connect to the CDB, the root, use local OS authentication or the root service name. For example, if you set the ORACLE_SID to the CDB instance name and use the command CONNECT / AS SYSDBA, you are connected to the root under common SYS user granted system privileges to manage and maintain all PDBs.

If you use the service name, you can use the EasyConnect syntax or the alias from the tnsnames.ora.

Using EasyConnect, you would enter the following connect string:

```
SQL> CONNECT username@hostname:portnumber/service_name
```

Using the tnsnames.ora file, you would enter the following connect string:

```
SQL> CONNECT username@net_service_name
```

To connect to a desired PDB, use either Easyconnect or the alias from the tnsnames.ora file, for example as shown in the slide. In our examples, the net service name in the tnsnames.ora matches the service name.

If your database is not being managed by Oracle Restart or Oracle Clusterware, you can create or modify services for each PDB using DBMS_SERVICE package. In this case, the PDB property is set to the current PDB where the operation is performed.

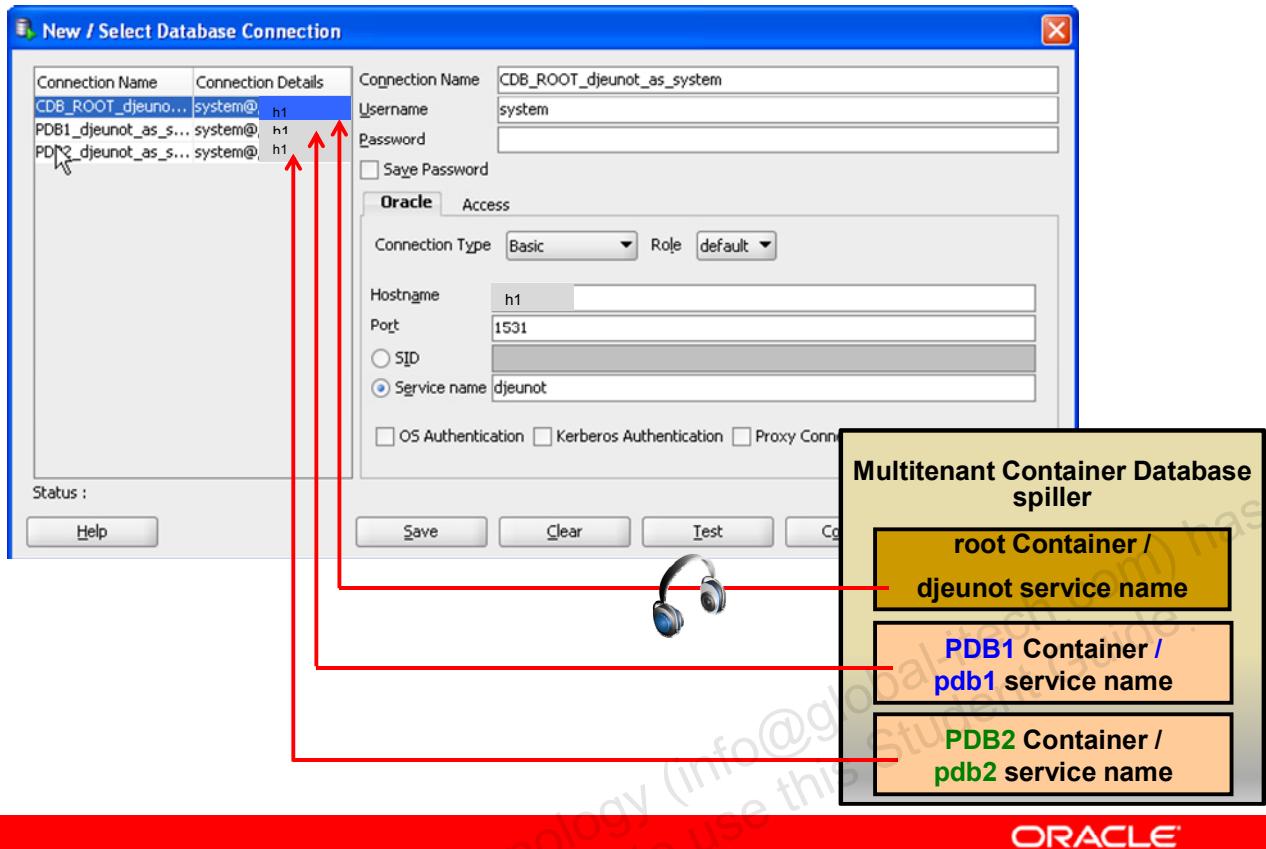
```
SQL> CONNECT system@salespdb
SQL> EXEC DBMS_SERVICE.CREATE_SERVICE('salesrep','salesrep')
SQL> EXEC DBMS_SERVICE.START_SERVICE('salesrep')
```

You can also create or modify services for each PDB using Server Control utility with `srvctl add service` command specifying the PDB name using an additional attribute `-pdb`:

```
$ srvctl add service -db mycdb -service salesrep -pdb salespdb
```

This example adds the salesrep service for the PDB salespdb in the CDB with `DB_UNIQUE_NAME mycdb`.

Connection with SQL*Developer

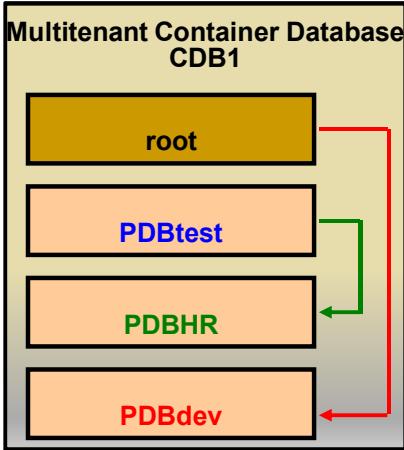


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can also use SQL Developer to connect to the CDB, to the root and to any of the PDBs. As shown in the slide, in the left pane, there are three configured connection names, one to connect to the root, another one to connect to `pdb1` and a third one to connect to `pdb2`.

In the right pane, the connection name `CDB_root_djeunot_as_system` shows that the connection connects to the service name `djeunot` being the CDB name—hence the root—as user `SYSTEM`.

Switching Connections



Two possible ways to switch connection between containers within a CDB:

- Reconnect:

```
SQL> CONNECT / AS SYSDBA
SQL> CONNECT local_user1@PDBdev
```

- Use ALTER SESSION statement:

```
SQL> CONNECT sys@PDBtest AS SYSDBA
SQL> ALTER SESSION SET CONTAINER=PDBHR;
SQL> SHOW CON_NAME
SQL> ALTER SESSION SET CONTAINER=CDB$ROOT;
```

- Using CONNECT allows connection under common or local user.
- Using ALTER SESSION SET CONTAINER allows connection under common user only granted new system privilege SET CONTAINER.
 - AFTER LOGON triggers do not fire.
 - Transactions are still pending after switching containers.



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

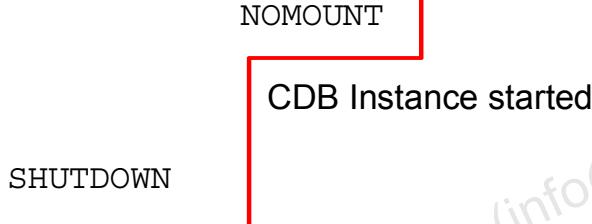
A CDB administrator can connect to any container in the CDB using either CONNECT or ALTER SESSION SET CONTAINER to switch between containers. For example, the CDB administrator can connect to root in one session, and then in the same session switch to the PDBHR container. This requires to be a common user known in both containers, and a system privilege SET CONTAINER.

- Using the command CONNECT allows connections under common or local users.
- Using ALTER SESSION SET CONTAINER allows connections under a common user only granted the new system privilege SET CONTAINER.
 - Be aware that AFTER LOGON trigger do not fire.
 - Transactions that are not committed nor rolled back in the original container are still in a pending state when switching to another container and when switching back to the original container.

Starting Up a CDB Instance

```
SQL> CONNECT sys@CDB1 AS SYSDBA  
SQL> STARTUP NOMOUNT
```

```
SQL> SELECT name, open_mode  
2 FROM v$pdbs;  
  
no rows selected
```



ORACLE

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

In a non-RAC environment, a CDB runs with a single instance. The instance is started exactly the same way as for non-CDB databases, using a STARTUP NOMOUNT statement. You need to be connected to the root of the CDB as SYSDBA to start the instance up.

Use V\$PDBS view to view the open mode of PDBs.

Mounting a CDB

```
SQL> CONNECT sys@cdb1 AS SYSDBA
SQL> STARTUP MOUNT
```

Or

```
SQL> ALTER DATABASE cdb1 MOUNT;
```

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

NAME	OPEN_MODE
PDB\$SEED	MOUNTED
PDB1	MOUNTED
PDB2	MOUNTED

NOMOUNT
Instance started
SHUTDOWN

MOUNT

- CDB control files opened for the instance
- Root mounted
- PDBs mounted

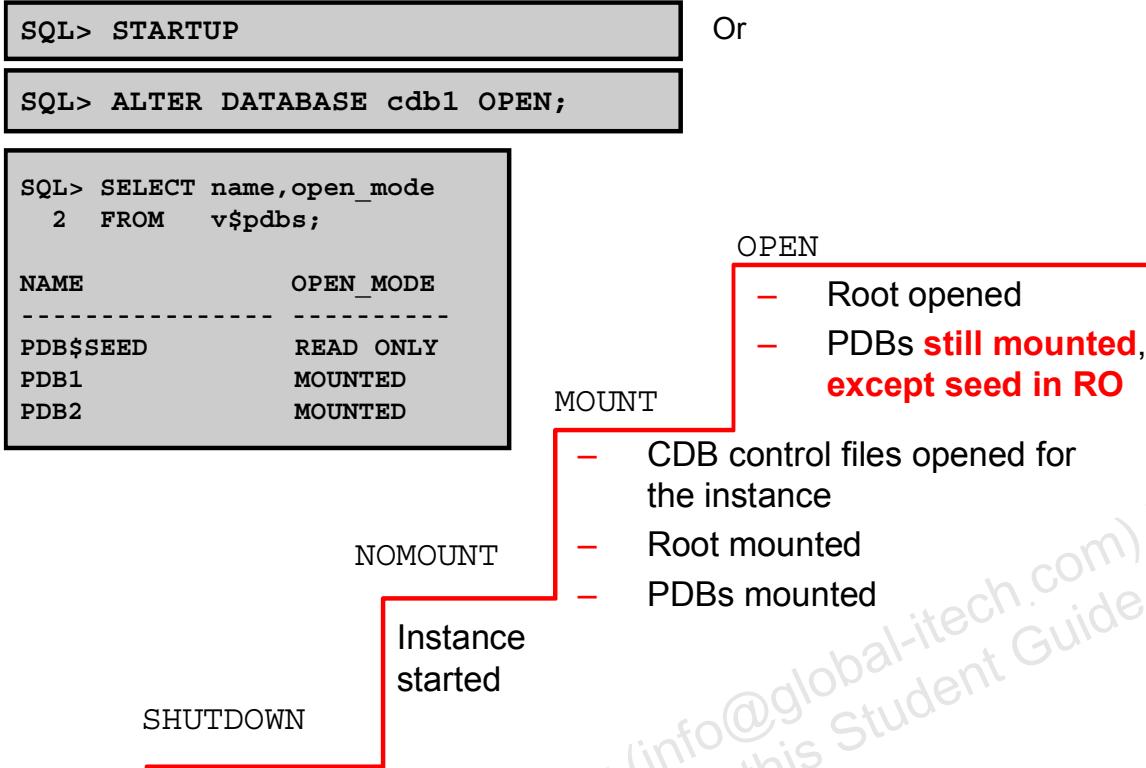
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A CDB may be started up with the **MOUNT** option. The command used to mount a CDB is the same as for non-CDB databases, using a **STARTUP MOUNT** statement. Again, you need to be connected to the root of the CDB as **SYSDBA** to perform this operation.

When a CDB is mounted, the root is mounted which means that the control files are opened, as well as the PDBs.

Use the **open_mode** column from the **V\$PDBS** view to verify that all PDBs are mounted.

Opening a CDB



ORACLE

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

When a CDB is opened, the root is opened which means that all redo log files and root data files are opened while all PDBs are still only mounted. Only connections to the root allow operations. Separate statements need to be issued to open PDBs, unless triggers automatically open PDBs after `STARTUP DATABASE`.

Use the `open_mode` column from the `V$PDBS` view to verify that all PDBs are still mounted, except the seed being in read-only mode. This allows creating new PDBs from it.

Opening a PDB

```
SQL> CONNECT sys@cdb1 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN;
```

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

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

NAME	OPEN_MODE
PDB\$SEED	READ ONLY
PDB1	READ WRITE
PDB2	READ WRITE

Or

PDB OPEN

PDBs **opened RW**,
except seed in RO

OPEN

- Root opened
- PDBs still mounted,
except seed in RO

MOUNT

- CDB control files opened for the instance
- Root mounted
- PDBs mounted

NOMOUNT

Instance
started

SHUTDOWN

ORACLE

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

To open a PDB or some of them or all of them, connect to the root as **SYSOPER** or **SYSDBA** and issue an **ALTER PLUGGABLE DATABASE OPEN** statement, specifying the PDB or PDBs names or **ALL EXCEPT** or **ALL**.

This operation opens the data files of the PDBs opened and provides availability to users.

Use the **open_mode** column from the **V\$PDBS** view to verify that all PDBs are all in **READ WRITE** open mode, except the seed being still in **READ ONLY** open mode.

You can also open a PDB while connected as **SYSDBA** within the PDB. In this case, it is not necessary to name the PDB to open.

Closing a PDB

```
SQL> CONNECT / AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pdb1
  2 CLOSE IMMEDIATE;
SQL> ALTER PLUGGABLE DATABASE
  2 ALL EXCEPT pdb1 CLOSE;
SQL> ALTER PLUGGABLE DATABASE
  2 ALL CLOSE;
```

PDB CLOSE

PDBs closed

```
SQL> CONNECT sys@pdb1 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE CLOSE;
Or
SQL> SHUTDOWN IMMEDIATE;
```

CDB OPEN

- Root opened
- PDBs mounted, except seed still RO

MOUNT

- CDB control files opened for the instance
- Root mounted
- PDBs mounted

NOMOUNT

SHUTDOWN

Instance started



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

To close a PDB or some of them or all of them, connect to the root as `SYSOPER` or `SYSDBA` and issue an `ALTER PLUGGABLE DATABASE CLOSE` statement, specifying the PDB or PDBs names or `ALL EXCEPT` or `ALL`. If you use the clause `CLOSE IMMEDIATE`, the transactions in the selected PDBs are rolled back and the sessions disconnected. If you omit the `IMMEDIATE` clause, the statement hangs until all sessions are disconnected.

This operation closes the data files of the closed PDBs and denies availability to users. Though all PDBs are closed, it is still possible to perform operations from the root, such as dropping PDBs or creating new PDBs from the seed.

The statement `SHUTDOWN IMMEDIATE` when connected to a PDB is equivalent to `ALTER PLUGGABLE DATABASE CLOSE`. It closes the PDB.

Note: Though `SHUTDOWN IMMEDIATE` issues the traditional message `ORACLE instance shut down`, this does not mean that the instance is down. Understand that the PDB is closed.

If the PDB is already closed, then the message explains the situation clearly with:

```
SQL> shutdown immediate
ORA-65020: Pluggable database already closed
```

Shutting Down a CDB Instance

```
SQL> CONNECT sys@CDB1 AS SYSDBA  
SQL> SHUTDOWN IMMEDIATE
```

- All PDBs closed (no new specific message)
- CDB closed
- CDB dismounted
- Instance shut down

```
SQL> CONNECT sys@PDB1 AS SYSDBA  
SQL> SHUTDOWN IMMEDIATE
```

- PDB closed



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

When a CDB is shut down, the data files of the root container and all PDBs are closed, then all the controlfiles are closed and in the last step the instance is shut down.

When a PDB is shut down, this means that the data files of the PDB are closed.

Database Event Triggers: Automatic PDB Opening

Trigger to automatically open PDBs after STARTUP:

- AFTER STARTUP → ON DATABASE

New database event triggers:

- AFTER CLONE → ON PLUGGABLE DATABASE
- BEFORE UNPLUG → ON PLUGGABLE DATABASE
 - Triggers are deleted after firing.
 - Any failure in AFTER CLONE or BEFORE UNPLUG trigger cause operation to fail.



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

After opening a CDB, the PDBs are by default kept in mounted mode. If you want the PDBs opened automatically AFTER STARTUP ON DATABASE, you can create an database event trigger that opens all PDBs after STARTUP.

```
CREATE TRIGGER Open_All_PDBs
    after startup on database
begin
    execute immediate 'alter pluggable database all open';
end Open_All_PDBs;
/
```

AFTER CLONE trigger fires in the copy of a PDB after it has been cloned. ON PLUGGABLE DATABASE must be specified when creating an AFTER CLONE trigger. Such triggers are deleted after they fire. Any failure in an AFTER CLONE trigger will cause the clone operation to fail.

BEFORE UNPLUG trigger fires in the PDB before any unplug operation starts. ON PLUGGABLE DATABASE must be specified when creating a BEFORE UNPLUG trigger. Such triggers are deleted after they fire. Any failure in a BEFORE UNPLUG trigger causes the unplug operation to fail.

Changing PDB Mode

After closing a PDB, open in:

- Restricted mode:

```
SQL> CONNECT sys@pdb1 AS SYSDBA  
SQL> ALTER PLUGGABLE DATABASE CLOSE;
```

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

```
SQL> SELECT name, open_mode FROM v$pdbs;  
  
NAME                      OPEN_MODE  
-----  
PDB1                      RESTRICTED
```

- Read-only mode:

```
SQL> CONNECT / AS SYSDBA  
SQL> ALTER PLUGGABLE DATABASE ALL OPEN READ ONLY;
```

- Read-write mode



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

You can change the mode of each PDB to perform specific administration operations.

The first example opens the PDB in the RESTRICTED mode. This allows only users with RESTRICTED SESSION privilege to connect. This allows the local administrator of the PDB to manage file movement, backups, and to prevent sessions from accessing the data.

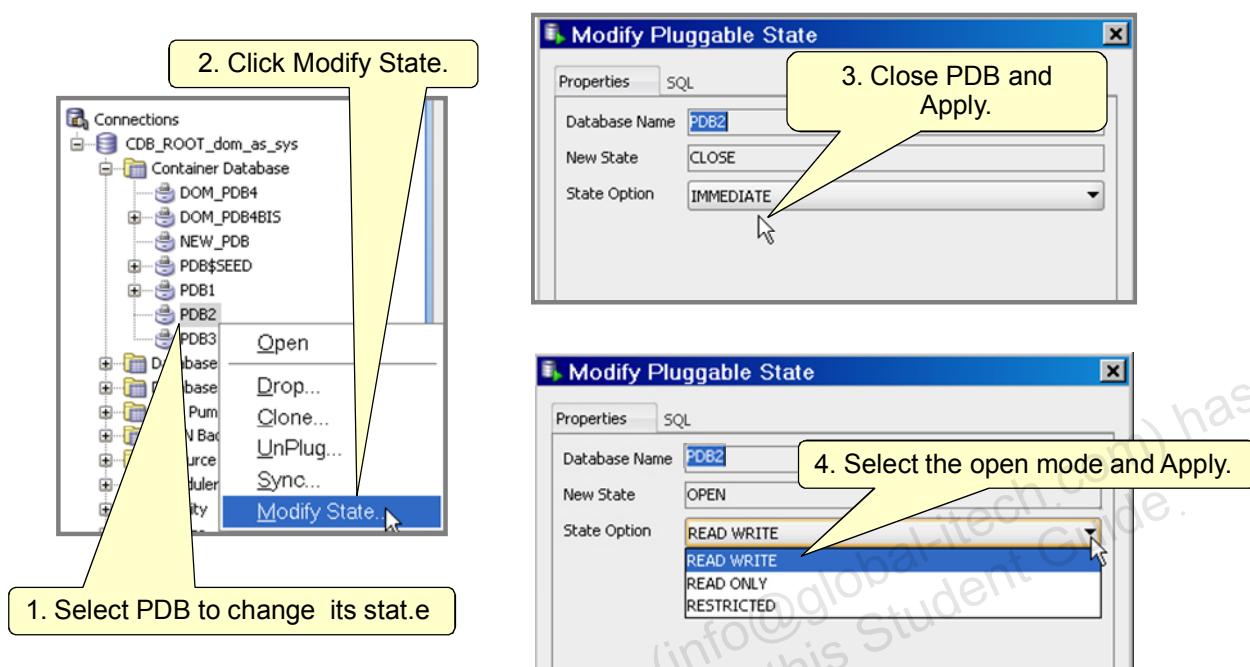
Use the V\$PDBS view to verify that the PDB is in RESTRICTED open mode.

The second example opens the PDB in a READ ONLY mode. Any session connected to the PDB can perform read-only transactions only.

To change the open mode, first close the PDB. You can apply the same open mode to all PDBs or to some of them.

Changing PDB Mode: With SQL Developer

Perform the same operations with SQL Developer.



ORACLE

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

If the source PDB is still opened, you have to open the PDB in READ ONLY mode. So, first close the PDB and then open the PDB in READ ONLY mode. Then you can reiterate the clone operation.

Modifying PDB Settings

- Bring a PDB data file online.
- Change the PDB default tablespace.
- Change the PDB default temporary tablespace.
- Set the PDB storage limit.
- Change the global name.

```
SQL> CONNECT sys@pdb1 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE
  2  DATAFILE '/u03/pdb1_01.dbf' ONLINE;
```

```
SQL> ALTER PLUGGABLE DATABASE DEFAULT TABLESPACE pdb1_tbs;
```

```
SQL> ALTER PLUGGABLE DATABASE DEFAULT TEMPORARY TABLESPACE
  2  temp_tbs;
```

```
SQL> ALTER PLUGGABLE DATABASE STORAGE (MAXSIZE 2G);
```

```
SQL> ALTER PLUGGABLE DATABASE RENAME GLOBAL_NAME TO pdbAPP1;
```



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

You can modify settings of each PDB without necessarily changing the mode of the PDB. You have to be connected in the PDB to perform the settings changes.

The first example uses a DATAFILE clause to bring the data file online.

The second example sets the default permanent tablespace to pdb1_tbs for the PDB.

The third example sets the default temporary tablespace to temp_tbs for the PDB.

The fourth example sets the storage limit for all tablespaces that belong to the PDB to two gigabytes.

The fifth example changes the global database name of the PDB to pdbAPP1. The new global database name for this PDB must be different from that of any container in the CDB and this operation can be done only in restricted mode.

Instance Parameter Change Impact

- A single SPFILE per CDB
- PDB value changes are:
 - Loaded in memory after PDB close
 - Stored in dictionary after CDB shutdown
 - Only for parameters `ISPDB_MODIFIABLE=TRUE`

```
SQL> CONNECT sys@pdb1 AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET ddl_lock_timeout=10;
System altered.
SQL> show parameter ddl_lock_timeout

NAME                                     TYPE        VALUE
-----                                     -----
ddl_lock_timeout                         boolean     10
```



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

There is a single SPFILE per CDB to store parameters. Values of parameters are associated with the root, apply to the root, and serve as default values for all other containers.

You can set different values in PDBs for the parameters where the column `ISPDB_MODIFIABLE` in `V$PARAMETER` is `TRUE`. These are set in the scope of a PDB, then they are remembered properly across PDB close/open and across bouncing the CDB instance. They also travel with clone and unplug/plug operations. Other initialization parameters can be set for the root only.

```
SQL> select PDB_UID, NAME, VALUES$ from pdb_spfile$;

PDB_UID      NAME                      VALUES$
-----      -----
3100074415  optimizer_use_sql_plan_baselines  FALSE
2862146267  optimizer_use_sql_plan_baselines  FALSE
2862146267  ddl_lock_timeout                40
```

Instance Parameter Change Impact: Example

```
SQL> CONNECT sys@pdb2 AS SYSDBA  
  
SQL> ALTER SYSTEM SET ddl_lock_timeout=20 scope=BOTH;  
  
SQL> ALTER PLUGGABLE DATABASE CLOSE;  
SQL> ALTER PLUGGABLE DATABASE OPEN;
```

```
SQL> CONNECT / AS SYSDBA  
SQL> select VALUE, ISPDB_MODIFIABLE, CON_ID  
  2  from V$SYSTEM_PARAMETER  
  3  where name ='ddl_lock_timeout';  
  
VALUE          ISPDB      CON_ID  
-----  
0              TRUE       0  
10             TRUE       3  
20             TRUE       4
```



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

In this example, a different value of the DDL_LOCK_TIMEOUT parameter is set in pdb2. The value changes are remembered after the PDB close and open. The new column CON_ID in view V\$SYSTEM_PARAMETER shows the DDL_LOCK_TIMEOUT value in each container, the root, pdb1 and pdb2.

Quiz

When you STARTUP a CDB, is the sequence of operations performed automatically? True or False ?

- a. The instance is started.
- b. Control files are opened.
- c. The root container is opened (redo logs and root data files).
- d. Seed pluggable database is in READ ONLY mode.
- e. Other PDBs are still in MOUNTED mode.
- f. Triggers may fire if they exist to open other PDBs.



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

Answer: True

Quiz

When you STARTUP a CDB, all PDBs can be opened in read-write mode.

- a. False
- b. True



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

Answer: a

Summary

In this lesson, you should have learned how to:

- Establish connections to CDB / PDB
- Start up and shut down a CDB
- Open and close PDBs
- Create event triggers to open PDBs
- Change the different modes and settings of PDBs
- Evaluate the impact of parameter value changes

Practice 4 Overview: Managing a CDB and PDBs

These practices cover the following topics:

- Starting up and shutting down a CDB
- Connecting to PDBs and displaying context
- Opening and closing PDBs



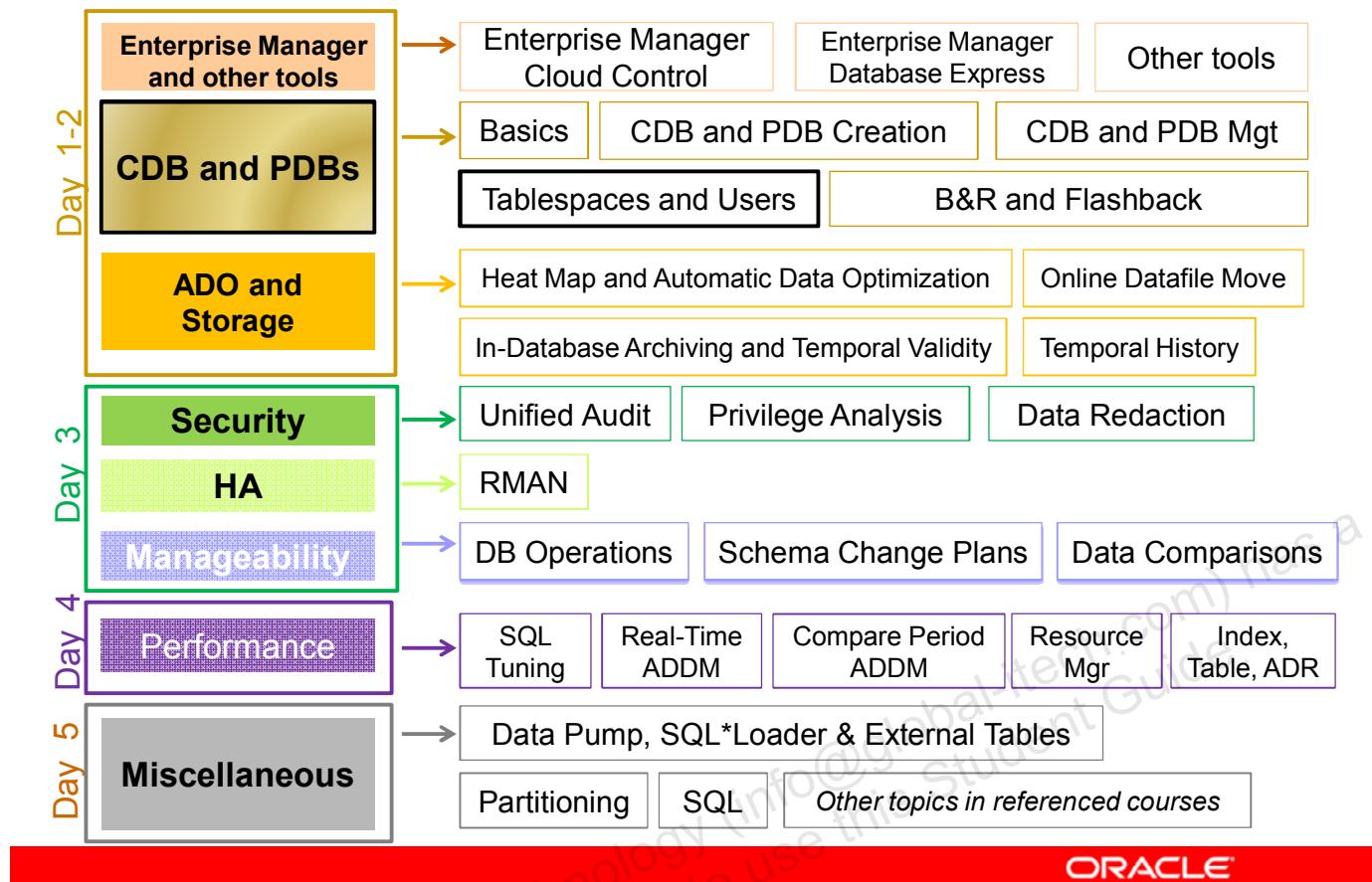
Managing Tablespaces and Users in CDB and PDBs



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.

Objectives

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

- Permanent tablespaces in CDB and PDBs
- Temporary tablespaces in CDB and PDBs
- Common and local users
- Common and local roles
- Common and local privileges



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

Note

For a complete understanding of Oracle PDB new feature and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Administrator's Guide 12c Release 1 (12.1)*
- *Oracle Database Security Guide 12c Release 1 (12.1)*

Refer to other sources of information:

- *Oracle Database 12c New Features Demo Series* demonstrations under Oracle Learning Library:
 - *CDB and PDBs Administration – Manage Common and Local Users*
 - *CDB and PDBs Administration – Manage Common and Local Roles and Privileges*

Tablespaces in PDBs

- A tablespace in a PDB can contain objects associated with exactly one PDB.
- In CREATE DATABASE:
 - `USER_DATA TABLESPACE` replaces automatic creation of `USERS` tablespace by DBCA.
- There is only one active `UNDO` tablespace per CDB.



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

In a non-CDB, all the tablespaces belong to one database. In the CDB, one set of tablespaces belong to the root container, and each PDB has its own set of tablespaces.

Common objects are created and their data stored in a tablespace in the root container. The common object is visible in the PDBs through links.

There are new clauses in the `CREATE DATABASE` command. The `USER_DATA TABLESPACE` allows you to specify a default tablespace other than `USERS` when using DBCA to create a database. This tablespace will also be used for XDB options.

The `UNDO` tablespace is common to all PDBs, that is, there is only one active `UNDO` tablespace per CDB.

Creating Permanent Tablespaces in a CDB

- Create a permanent tablespace in the root container:

```
SQL> CONNECT system@cdb1
SQL> CREATE TABLESPACE tbs_CDB_users DATAFILE
  2  '/u1/app/oracle/oradata/cdb/cdb_users01.dbf'
  3  SIZE 100M;
```

- Create a permanent tablespace in a PDB:

```
SQL> CONNECT system@PDB1
SQL> CREATE TABLESPACE tbs_PDB1_users DATAFILE
  2  '/u1/app/oracle/oradata/cdb/pdb1/users01.dbf'
  3  SIZE 100M;
```

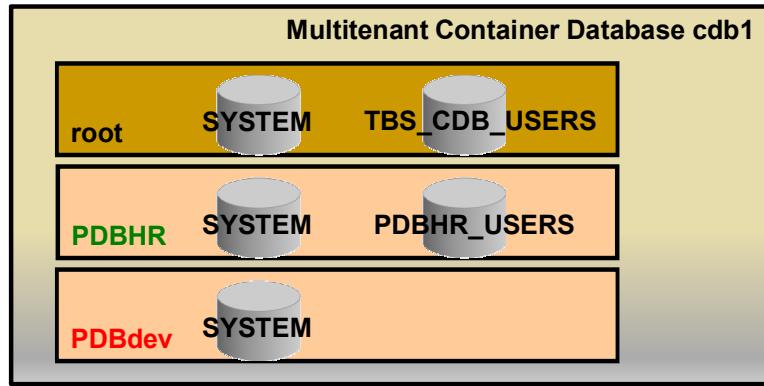


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `CREATE TABLESPACE` command should be familiar. The change in its behavior in a CDB is that the tablespace is created in the container where the command is executed.

Separating the data files into different directories by PDB can help determine which files belong to which PDB, though it is not necessary.

Assigning Default Tablespaces



- In the CDB:

```
SQL> CONNECT system@cdb1
SQL> ALTER DATABASE
  2> DEFAULT TABLESPACE tbs_CDB_users;
```

- In the PBD:

```
SQL> CONNECT pdb1_admin@pdbhr
SQL> ALTER PLUGGABLE DATABASE
  2> DEFAULT TABLESPACE pdbhr_users;
```

ORACLE

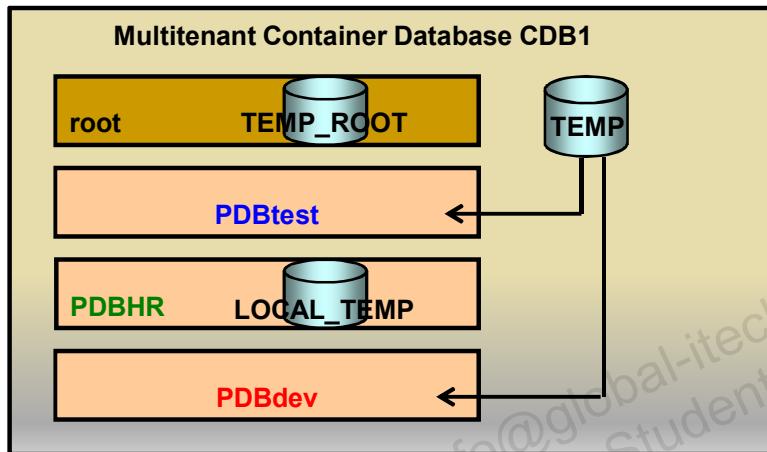
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The default tablespace for a database is a database property. To change the default tablespace for a CDB root container you must: connect to the root container as a user with the proper privileges, and issue the `ALTER DATABASE` command. This operation does not change the default permanent tablespace of PDBs.

To change the default tablespace for a PDB you must connect to the PDB as a user with the proper permissions and issue the `ALTER PLUGGABLE DATABASE` command. When connected to the PDB, the `ALTER DATABASE` and `ALTER PLUGGABLE DATABASE` commands perform the same modifications to the PDB. The `ALTER DATABASE` command is allowed for backward compatibility.

Creating Local Temporary Tablespaces

- Only one default temporary tablespace or tablespace group is allowed per CDB or PDB.
- Each PDB can have temporary tablespaces or tablespace groups.



ORACLE

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

A CDB can have only one default temporary tablespace or tablespace group. As with a non-CDB there can be other temporary tablespaces to which users can be assigned. The default temporary tablespace is a shared resource for all PDBs, that is any user in any PDB will use the CDB default temporary tablespace unless that user is assigned to a specific temporary tablespace.

PDBs can have temporary tablespaces for use by users in the PDB. These temporary tablespaces will be transported with the PDB when it is unplugged.

Assigning Default Temporary Tablespaces

- In the CDB:

```
SQL> CONNECT system@cdb1
SQL> ALTER DATABASE
  2> DEFAULT TEMPORARY TABLESPACE temp_root;
```

- In the PBD:

```
SQL> CONNECT pdb1_admin@pdbhr
SQL> ALTER PLUGGABLE DATABASE
  2> DEFAULT TEMPORARY TABLESPACE local_temp;
```

or

```
SQL> CONNECT pdb1_admin@pdbhr
SQL> ALTER DATABASE
  2> DEFAULT TEMPORARY TABLESPACE local_temp;
```



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

The default temporary tablespace for the CDB is set at the root container level. There is one default temporary tablespace (or tablespace group) for an entire CDB. There may be multiple temporary tablespaces, but only one can be the default.

A default temporary tablespace (or tablespace group) can be set for each PDB. A PDB may have multiple temporary tablespaces, but only one default per PDB.

When you create a user you can specify a temporary tablespace to be used by that user. If a temporary tablespace is not specified, the default tablespace for the PDB is used. If a default tablespace has not been specified for the PDB, the temporary tablespace for the CDB is used.

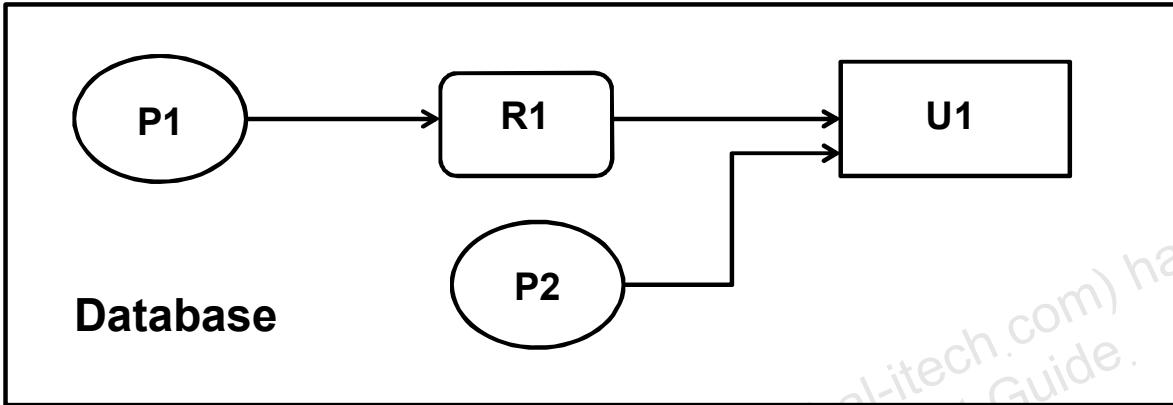
The default temporary tablespace in the CDB is shared by all the PDBs. The amount of space a single PDB can use in the shared temporary tablespace can be set in the PDB by:

```
ALTER PLUGGABLE DATABASE STORAGE (MAX_SHARED_TEMP_SIZE 500M);
```

When you unplug a PDB from a CDB, its temporary tablespaces are also unplugged.

Users, Roles, and Privileges

- Each user can exercise granted privileges in the context of a single database.
- A role is a collection of privileges.



ORACLE

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

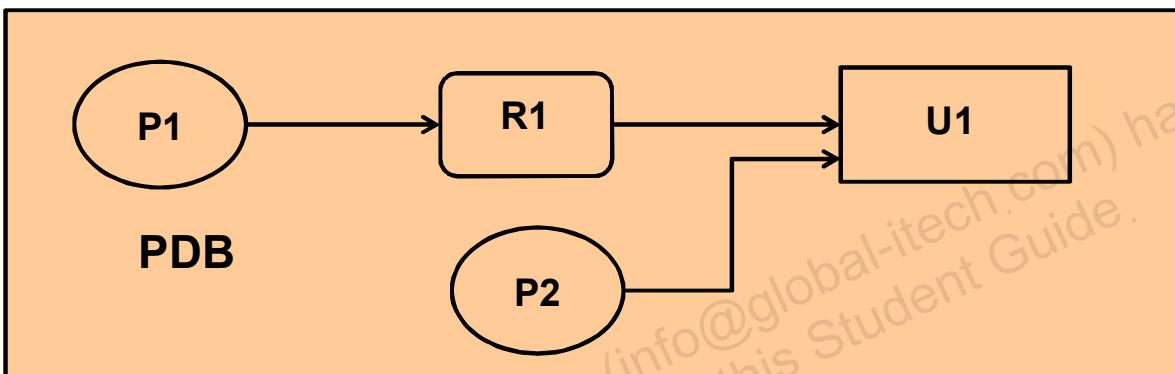
In a non-CDB, a user can perform operations authorized by system privileges, on objects allowed by object privileges. These privileges apply only to the database to which the user is connected.

A role is a set of privileges. The privileges can be exercised in the context of a specific database. Privileges can be granted either directly or through roles.

In the past, the phrase “in the context of a single database” has been implied. When you consider that there was only one database per instance, this seems a trivial assertion, but this is key to understanding local users, local privileges, and local roles in PDBs.

Local Users, Roles, and Privileges

- Each **local** user can exercise granted privileges in the context of a single **PDB**.
- A **local** role is a collection of privileges that are assigned at user login to a specific **PDB**.
- A **local** privilege is one that is granted in the context of a single **PDB**.



ORACLE

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

A local user exists in one and only one PDB. Even if multiple PDBs have local users with the same name and credentials, each local user is distinct.

In a PDB, the local users behave in exactly the same manner as users defined in the context of a non-CDB. A local user can perform DDL operations authorized by system privileges and DML and other operations on objects allowed by object privileges. The privileges apply only to the PDB in which the local user is defined. The differences between the rules for a non-CDB and a PDB are highlighted. As you can see, they are almost identical.

The commands to create local users and roles in a PDB are the same as for a non-CDB.

Creating a Local User

- A local user with proper privileges can create another local user.
- The syntax is the same as in non-CDB.

```
SQL> CREATE USER GEORGE IDENTIFIED BY x;
```

- You cannot create local users in the root.



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

A local user can be created by another local user with the proper privilege, typically one with the DBA role in the PDB. The CREATE USER privilege is required.

The syntax is the same as in a traditional or non-CDB. This is in keeping with the application transparency goal for PDBs. The syntax for commands in a PDB are identical to commands in a non-CDB.

Note: The DROP USER and ALTER USER commands are identical, as the commands in a non-CDB.

You cannot create local users in the root.

Common Users

Multitenant Container Database CDB1

The screenshot shows three database instances: 'root', 'PDB_HR', and 'PDB_SALES'. In the 'root' instance, a table named 'CDB_USERS' is displayed with the following data:

USERNAME	COMMON
SYS	YES
SYSTEM	YES
C##DBA	YES
DBSNMP	YES
HR_MGR	NO
SALES	NO

In the 'PDB_HR' instance, a table named 'DBA_USERS' is displayed with the following data:

USERNAME	COMMON
SYS	YES
SYSTEM	YES
C##DBA	YES
DBSNMP	YES
HR_MGR	NO

In the 'PDB_SALES' instance, a table named 'DBA_USERS' is displayed with the following data:

USERNAME	COMMON
SYS	YES
SYSTEM	YES
C##DBA	YES
DBSNMP	YES
SALES	NO

A text box on the right states: "A common user can only be created in the root container."

A common user is a user that has the same username and authentication credentials across multiple PDBs, unlike the local user which exists in one and only one PDB.

A common user cannot have the same name as any local user across all of the PDBs. A common user can be created in the root container, and only in the root container: a common user is a database user that has the same identity in the root and in every existing and future PDB.

A local user cannot create a common user.

A common user with the proper privileges can create a common user using the CONTAINER=ALL clause. To create a common user, common user must have the SET CONTAINER privilege for all the PDBs. A common user can connect to a specific PDB and with the proper privilege in the PDB create a local user with the CONTAINER = CURRENT clause.

Note: If a PDB is closed, the common and local users of the PDB are not visible because the metadata is retrieved from the PDB SYSTEM tablespace.

Creating a Common User

- A common user can create common users and local users.
- The **CONTAINER** clause determines the type of user created.

Create a **common** user in the root container:

```
SQL> CREATE USER C##_GEORGE IDENTIFIED BY x
  2 CONTAINER=ALL;
```

Create a **local** user in a PDB:

```
SQL> CREATE USER L_FRED IDENTIFIED BY y
  2 CONTAINER=CURRENT;
```



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

Only a common user with the `CREATE USER` and the `SET CONTAINER` common privileges can create a common user. A common privilege is a privilege granted across all the PDBs.

A common user with the `CREATE USER` and `SET CONTAINER` privilege in the PDB can create a local user in that PDB.

The type of user created is determined by the `CONTAINER` clause.

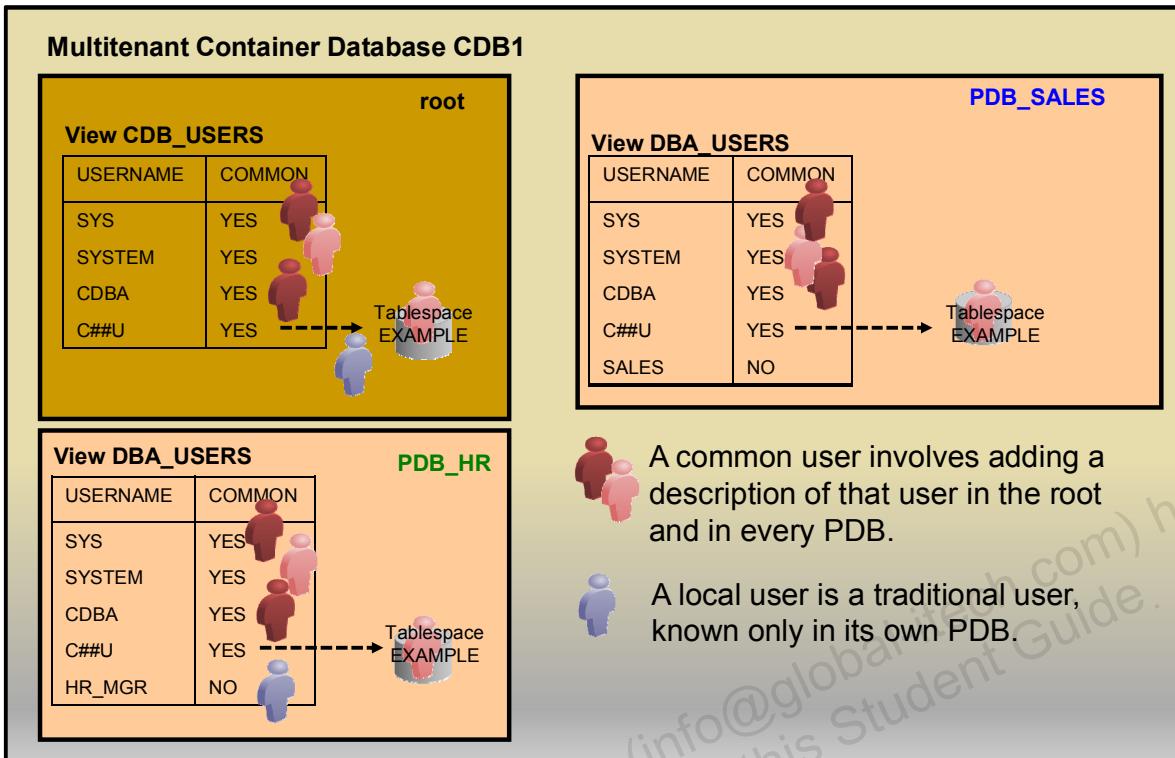
- The first command shown in the slide creates a common user when issued in the root container. To create a common user, you must be connected to the root container. The name of a common user must begin with `C##` characters.
- The second command creates a local user when the common user is connected to a specific PDB.

If the container clause is omitted, the default depends on the context. If the common user is connected to the root container the default is `CONTAINER=ALL`. If the common user is connected to a PDB the default is `CONTAINER=CURRENT`.

When creating a common user, any tablespace, tablespace group or profile specified in the `CREATE` command must exist in every PDB. If none of these are specified, the default `TABLESPACE`, `TEMPORARY TABLESPACE`, and `PROFILE` for the PDB will be used.

Note: the `DROP USER` and `ALTER USER` commands are the same as in a non-CDB.

Common and Local Schemas / Users



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

Adding, altering, or dropping a common user involves adding, changing or dropping a description of that user in the root and in every PDB. After changing the container, common users can or cannot perform operations depending on the type of privileges granted in the container (PDB).

A PDB that was not OPEN READ WRITE or RESTRICTED when a common user was created attempts to add the description of that common user as a part of the synchronization operation. If during the process, a new common username conflicts with a local username defined in a given PDB or a tablespace, a tablespace group or a profile which was specified in CREATE USER statement does not exist in that PDB, an error is reported.

Altering or dropping a common user involves modifying or removing a description of that user in the root container and in every PDB that is OPEN READ WRITE or RESTRICTED, effectively replaying the statement in each container.

Common and Local Privileges

- A privilege granted across all containers is a common privilege.
- A privilege granted in the context of a single PDB is a local privilege.
- Local users can exercise privileges only locally in the context of the PDB.
- Common users can exercise privileges only in the context of the PDB to which they are connected.
- Common users connected to the root container can exercise cross-container privileges, such as creating a common user.



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

A privilege becomes common or local based on the way it is granted. A privilege granted across all containers is a common privilege. A privilege granted in a specific PDB is local.

Common and local users can exercise common and local privileges that have been granted in the context of the PDB to which they are connected. A common user can be granted both common and local privileges. That means the privileges granted to a common user can be different in every PDB.

Only common users can connect to the root container. Even in the root container, privileges can be granted as local or common. Cross-container operations such as creating a common user can only be performed while connected to the root container and by common users with common privileges for those operations.

Granting and Revoking Privileges

- Grant a privilege commonly (becomes a common privilege)

```
SQL> GRANT drop user TO c##_user CONTAINER=ALL;
```

- Grant a privilege locally (becomes a local privilege).

```
SQL> GRANT alter user TO local_user CONTAINER=CURRENT;
```

- Grant a local privilege.

```
SQL> GRANT select any table TO local_user;
```

- Revoke a common privilege.

```
SQL> REVOKE drop user FROM c##_user CONTAINER=ALL;
```

- Revoke a local privilege.

```
SQL> REVOKE alter user FROM local_user;
```



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

The syntax and behavior of the GRANT and REVOKE are mostly unchanged. The syntax has been extended with the CONTAINER clause so that users with the proper privileges can grant privileges commonly or locally.

When a user grants a privilege with the CONTAINER=ALL clause, the privilege becomes a common privilege. The user must have the SET CONTAINER privilege for all the PDBs because he grants the privilege to the same common user in each container.

When a user grants a privilege with the CONTAINER=CURRENT clause, the privilege becomes a local privilege.

To grant a system privilege, the grantor must have the GRANT ANY PRIVILEGE system privilege or have been granted the privilege WITH ADMIN OPTION.

To grant an object privilege, the grantor must be the owner of the object, have the GRANT ANY OBJECT PRIVILEGE, or have the object privilege WITH GRANT OPTION.

The command to revoke privileges follow the same rules as grants. A common role cannot be revoked from a common user without the CONTAINER=ALL clause.

Object privilege grants track the grantor. If the grantor's privilege has the WITH GRANT OPTION, and is revoked so are the grants that the grantor made.

Creating Common and Local Roles

- A local user can create local roles.

```
SQL> CREATE ROLE L_HR CONTAINER=CURRENT;
```

- A common user can create common or local roles.

```
SQL> CREATE ROLE C##_R1 CONTAINER=ALL;
```

- Local roles can be granted to local or common users.
- Common roles can be granted to local or common users.
- Local roles can be granted to common roles.
- Common roles can be granted to local roles.



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

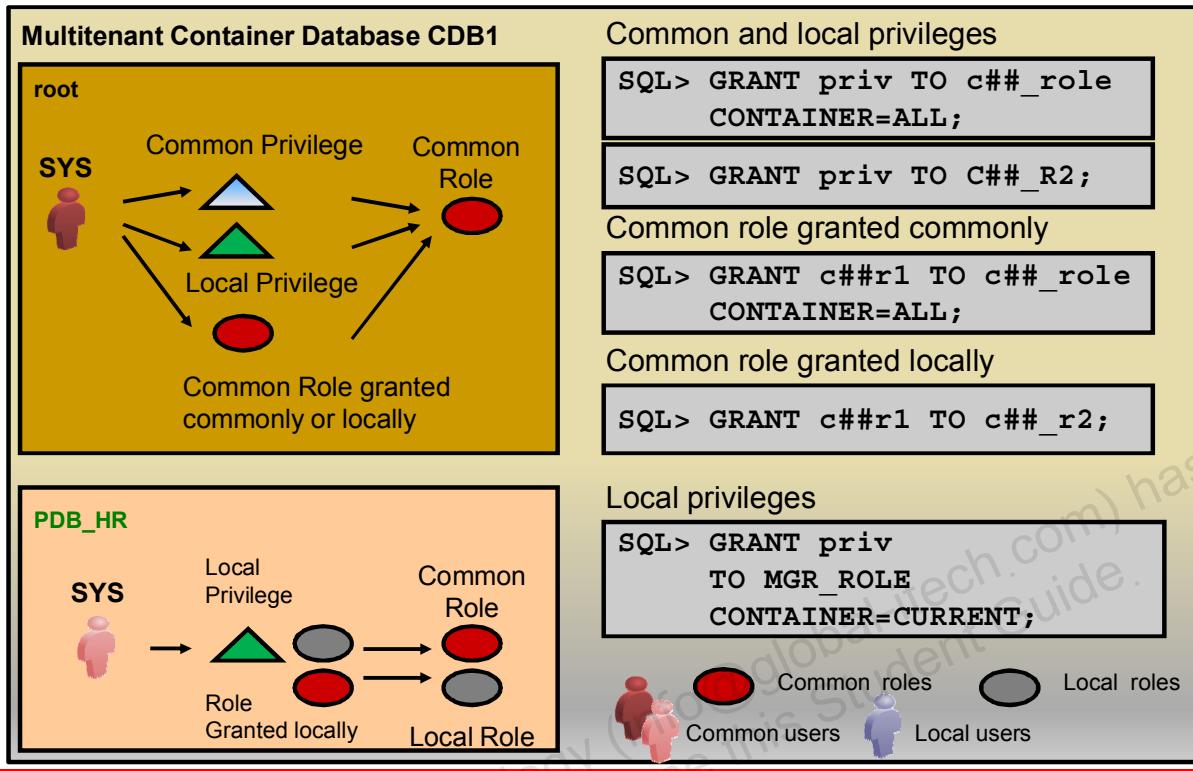
The syntax and privileges are the same as in a non-CDB for a local user to create a local role.

A user can create common roles if the user has the CREATE ROLE privilege, and SET CONTAINER for all PDBs. A common user can create a local role if the user has the CREATE ROLE and SET CONTAINER privilege for that PDB. The CONTAINER clause determines whether the role is common or local. A common role must begin with C## characters.

Any role can be granted to any role or user. It does not matter whether the user or role is defined to be local or common. Consider that a role is a container for privileges. When a role is granted to a user, the privileges in the role are limited to the context of the PDB.

For example, the common role C##_R1 is created with a common privilege CREATE SESSION. When the role C##_R1 is granted to a common user, C##_TEST, that user can connect to any PDB. But when C##_R1 is granted to a local user, lu_PDB1, defined in PDB_HR that local user can only connect to PDB_HR.

Granting Common or Local Privileges / Roles to Roles



ORACLE

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

To grant a role, the grantor must have the GRANT ANY ROLE privilege or have been granted the role WITH ADMIN OPTION.

Privileges assigned to a common role are common privileges as long as CONTAINER=ALL is specified and therefore are applied in all containers. If CONTAINER=ALL is not specified, the privilege assigned to a common role is local. The first command shown on the slide shows a privilege granted commonly to a common role, and therefore usable in each of the PDBs. The second command shown on the slide shows a privilege granted locally to a common role, and therefore usable only in the PDB where the command is executed.

Common roles can be created in the root container only.

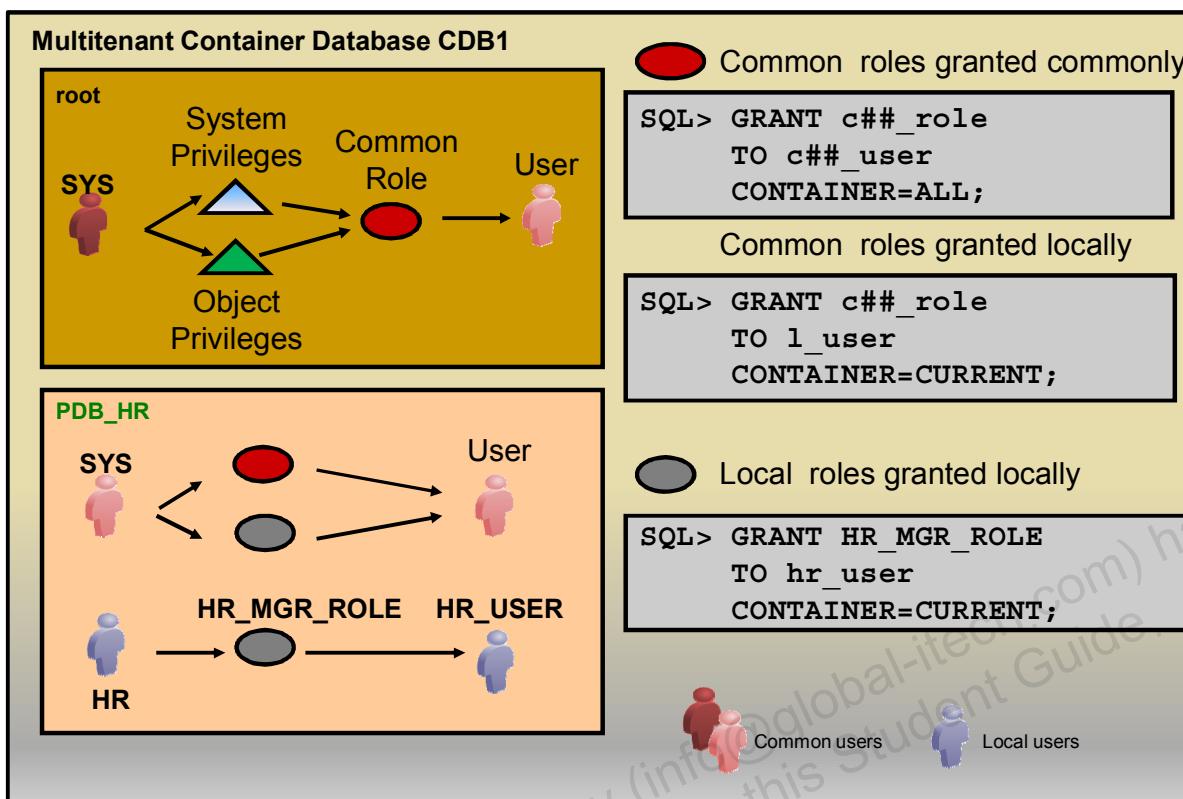
Example: A common role, C##_VIEWER, granted commonly the SELECT ANY TABLE privilege is granted to a common user C##_HR_CLERK. The SELECT ANY TABLE privilege can be used by C##_HR_CLERK while connected to any PDB.

Privileges assigned to local roles can only be local privileges. A local role can be created in a specific PDB and cannot be created in the root container. The fifth command executed in the PDB_HR container shows a privilege being granted locally to MGR_ROLE role.

Common roles may be granted to any user or role commonly or locally in the root. Local and common roles can be granted to any user or role locally only in any PDB.

A local user can be assigned a common role, and the privileges assigned by that role can only be exercised in the PDB where the local user is defined.

Granting Common and Local Roles to Users



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

Common roles may be granted to any user or role in any PDB. Local roles may be granted to any user or role that exists in the same PDB. In the first example the common role, C##_ROLE, is granted to the common user C##_USER. This will grant the role in all the containers.

A local role can be granted to a common user and a common role to a local user. A privilege granted directly or through a role is exercised in the context of a single PDB. Example: a common role C##_VIEWER, containing SELECT ANY TABLE privilege is granted to a common user C##_HR_CLERK. The SELECT ANY TABLE privilege can be used by C##_HR_CLERK while connected to any PDB. A local role LR_HR in the PDB_HR PDB has the INSERT INTO HR.EMPLOYEES privilege and is granted to C##_HR_CLERK. C##_HR_CLERK can use the INSERT INTO HR.EMPLOYEES privilege only while connected to PDB_HR.

A local user can be assigned a common role, and as before the privileges assigned by that role can only be exercised in the PDB where the local user is defined. If the local role LR_HR is granted to the C##_USER, the C##_USER would be able to exercise the INSERT INTO HR.EMPLOYEES privilege only while connected to PDB_HR.

Granting and Revoking Roles

- Grant a common role commonly.

```
SQL> GRANT c##_role TO c##_user CONTAINER=ALL;
```

- Grant a local role to a local user locally.

```
SQL> GRANT local_role TO local_user CONTAINER=CURRENT;
```

- Grant a local role to a common user locally.

```
SQL> GRANT l_role TO c##_user CONTAINER=CURRENT;
```

- Grant a common role to a common user locally

```
SQL> GRANT c##_role TO c##_user CONTAINER=CURRENT;
```

- Revoke a common role commonly.

```
SQL> REVOKE c##_role FROM c##_user CONTAINER=ALL;
```

- Revoke a local or common role locally.

```
SQL> REVOKE local_role FROM local_user;
```

```
SQL> REVOKE c##_role FROM c##_user  
2 CONTAINER=CURRENT;
```

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

The syntax and behavior of the GRANT and REVOKE are mostly unchanged. The syntax has been extended with the CONTAINER clause so that common and local roles can be granted or revoked commonly or locally.

To grant a role, the grantor must have the GRANT ANY ROLE privilege or have been granted the role WITH ADMIN OPTION.

A common role can be revoked locally from a user or role with the CONTAINER=CURRENT clause or commonly with the CONTAINER=ALL clause.

Creating Shared and Non-Shared Objects

- Non-shared tables can be created by common and local users.
- Shared tables CANNOT be created by common users; they are created by the customer.
- Shared tables can be created by Oracle-supplied users.



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

Non-shared tables and other non-shared database objects may be created in a PDB by either common or local users. Shared tables can be created by Oracle-supplied common users. Common users created by a user who is granted the CREATE USER privilege cannot create shared tables or objects.

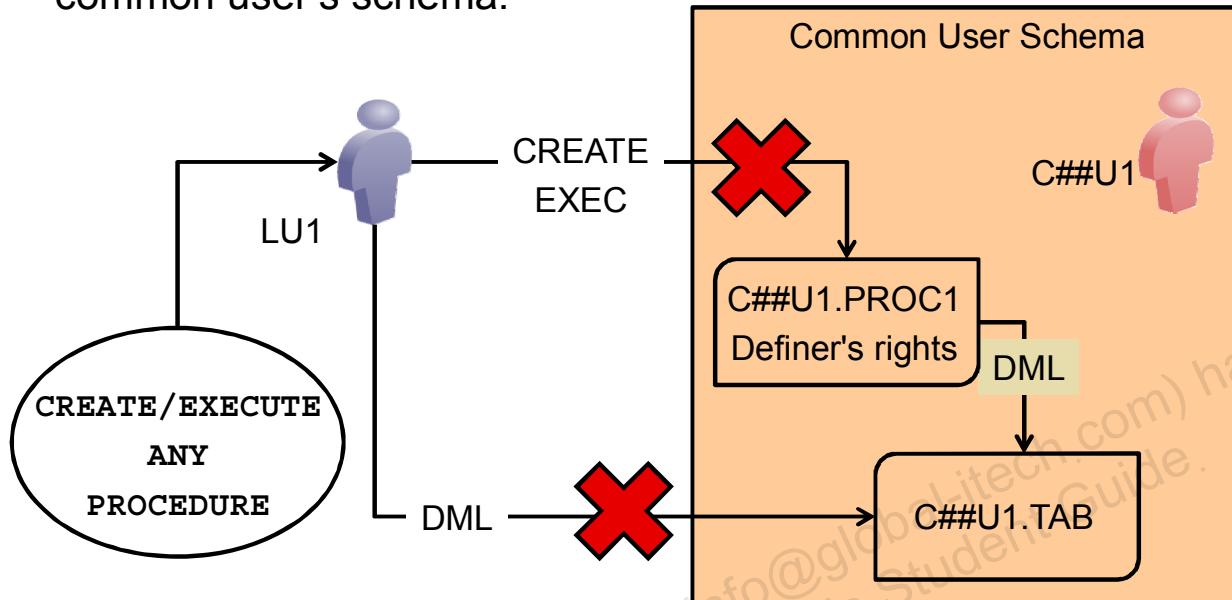
There are two types of shared objects, Object Link and Metadata Link.

- Object Links point to objects in the root and are used to share data across containers. An example of these Object Link objects is AWR objects.
- Metadata Links share metadata with objects in the root but have private copies of data. An example of these Metadata Link objects is DBA_xxx views.

You can find the type of sharing in SHARING column of DBA_OBJECTS.

Restriction on Definer's Rights

Local users **cannot** exercise local system privileges on a common user's schema.



ORACLE

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

A user cannot exercise local system privileges on a common user's schema. An example: Suppose a local user does not have privileges to perform DML on a table in a common user's schema, but does have CREATE ANY PROCEDURE and EXECUTE ANY PROCEDURE privileges. The local user is not allowed to create a definer's rights PL/SQL procedure that can issue statements as the common user against the common user's schema and then proceed to execute that procedure to get around restrictions imposed on local users.

Quiz

You can create roles in PDBs. Which statement is true about common and local roles?

- a. You can create local roles in the root.
- b. You can create common roles only when connected in the root.
- c. You can create common roles in PDBs.
- d. You can create common and local roles in seed.



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

Answer: b

Quiz

Choose all that apply to make a true statement. In a PDB, a common user and a local user can:

- a. Have the same name
- b. Be granted the same privileges
- c. Be granted both local and common roles
- d. Open and close the PDB if granted the appropriate privilege



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

Answer: b, c, d

Summary

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

- Any type of tablespaces in CDB and PDBs
- Common and local users
- Common and local roles
- Common and local privileges

Practice 5 Overview: Managing Tablespaces and Users in CDBs and PDBs

These practices cover the following topics:

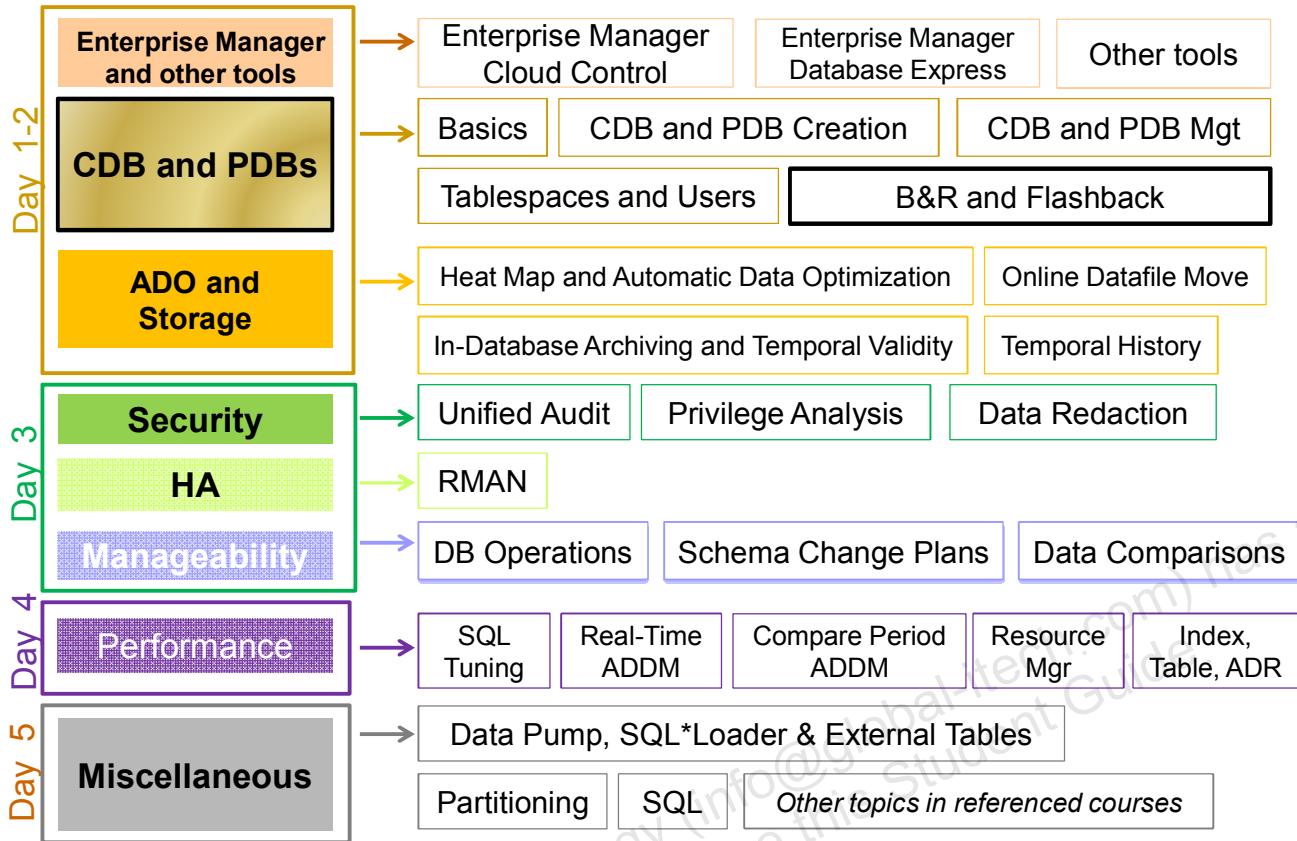
- Creating a tablespace in the root
- Creating a tablespace in a PDB
- Creating a common user
- Creating a local user
- Granting common and local privileges
- Creating common and local roles

Backup, Recovery, and Flashback CDBs and PDBs

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.

Objectives

After completing this lesson, you should be able to:

- Perform CDB and PDB backups
- Recover CDB from essential file loss
- Recover PDB from PDB data file loss
- Perform flashback database



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

Note

For a complete understanding of Oracle PDB new feature and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Administrator's Guide 12c Release 1 (12.1)*
- *Oracle Database Backup and Recovery User's Guide 12c Release 1 (12.1)*

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

- *Oracle Database 12c New Features Demo Series* demonstrations:
 - *How to Back Up and Recover CDB*
- *Oracle By Example (OBE)*:
 - *Performing a Point-In-Time Recovery for a Pluggable Database*

Goals

Back up CDB and applications independently:

- ARCHIVELOG mode at CDB level
- CDB backups and PDB backups
- Hot backup at CDB and PDB level

Recover CDB or PDBs at different levels:

- Instance failure: CDB level
- Complete media recovery:
 - CDB or PDB temp file
 - Control file / redo log file / root data file: CDB mounted
 - PDB data file: PDB opened if non SYSTEM data file
 - PDB data file: CDB mounted if SYSTEM data file
- Incomplete media recovery: CDB mounted or PDB closed
- Flashback database: CDB mounted



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

Any database, non-CDB or CDB, need to be backed up in case of potential recovery required. The ARCHIVELOG mode can be set only at CDB level and not at PDB level due to redo log files sharing.

In Oracle Database 11g, you can perform whole and partial non-CDB backup and on another level, hot tablespace or data file backups.

In Oracle Database 12c, you can still perform the same types of backups. The new level for backing up data is the PDB level.

The granularity of media recovery is very flexible and can be done for the entire CDB, for a PDB, for a tablespace, for a data file, or even for a block.

You can perform different types of recovery when a failure occurs:

- What to do if an instance failure occurs?
- What to do if a PDB temp file is missing?
- How to proceed if a control file is missing or corrupted?
- What to do if a root data file is missing or corrupted?
- How to proceed if a PDB data file is missing or corrupted?
- Do you use flashback database if a local schema is dropped or is there another type of flashback available at the PDB level?

New Syntax and Clauses in RMAN

```
$ export ORACLE_SID=cdb1
$ rman TARGET /
```

- DATABASE keyword operates on all PDBs and root:

```
RMAN> BACKUP DATABASE;
RMAN> RECOVER DATABASE;
```

- PDB operates on individual PDBs:

```
RMAN> BACKUP PLUGGABLE DATABASE hr_pdb, sales_pdb;
RMAN> RECOVER PLUGGABLE DATABASE hr_pdb;
```

- Backup, restore, recover the root using CDB\$ROOT keyword:

```
RMAN> BACKUP PLUGGABLE DATABASE "CDB$ROOT";
```

- Qualify tablespace of PDB with PDB name:

```
RMAN> BACKUP TABLESPACE sales_pdb:tbs2;
RMAN> RESTORE TABLESPACE system;
```



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

You can use Recovery Manager (RMAN) or Enterprise Manager to back up and recover entire CDBs, partial CDBs, individual whole PDBs, or partial PDBs such as tablespace/datafile of specific PDBs.

The CDB or PDBs are the possible targets for RMAN; an individual PDB is a valid RMAN TARGET database. Connect to the root or a PDB as a user with SYSDBA or SYSBACKUP privilege.

The traditional syntax such BACKUP DATABASE, RESTORE DATABASE, RECOVER DATABASE operates on the root and all of its PDBs, and, therefore, on the whole CDB, or on a single PDB depending on the connection.

A new syntax, namely PDB, is introduced to allow backup, restore, recover single or several PDBs.

To back up, restore, or recover the root, CDB\$ROOT must be used.

RMAN TABLESPACE syntax can be qualified with the PDB name so that user can specify the name of the TABLESPACE as it is known to PDB during backup, restore, recover commands. If no PDB qualifier is used, then root is used by default. To list the tablespaces and their associated PDB, use REPORT SCHEMA syntax.

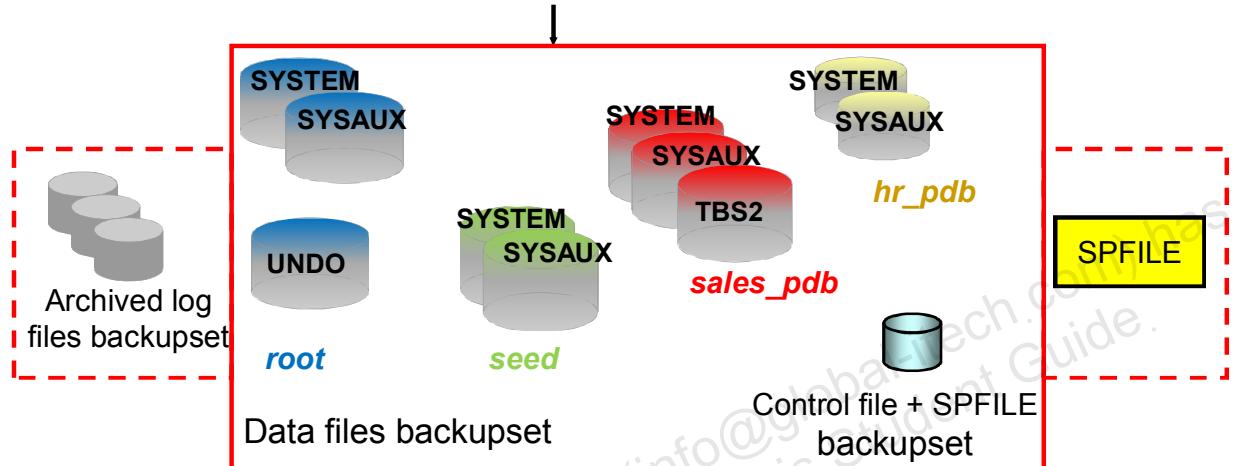
The same new clauses can be applied to DUPLICATE, SWITCH, REPORT, CONVERT, CHANGE, LIST, DELETE.

The root needs to be available for recovery, backup, and RMAN backups.

CDB Backup: Whole CDB Backup

- Back up all the PDB's data files and root files.

```
RMAN> CONNECT TARGET /
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```



ORACLE

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

RMAN can back up the entire CDB and contained or individual PDBs. In addition, individual tablespaces or data files can be backed up from a specific PDB.

A whole database backup of a CDB can be, in a similar way to non-CDBs, either backup sets or image copies of the entire set of data files, namely the root data files and all PDBs data files, and the control file. You can optionally include the server parameter file (SPFILE) and archived redo log files.

Using RMAN to make an image copy of all the CDB files simply requires mounting or opening the CDB, starting RMAN, connecting to the root with SYSDBA or SYSBACKUP privilege, and entering the BACKUP command shown in the slide. Optionally, you can supply the DELETE INPUT option when backing up archivelog files.

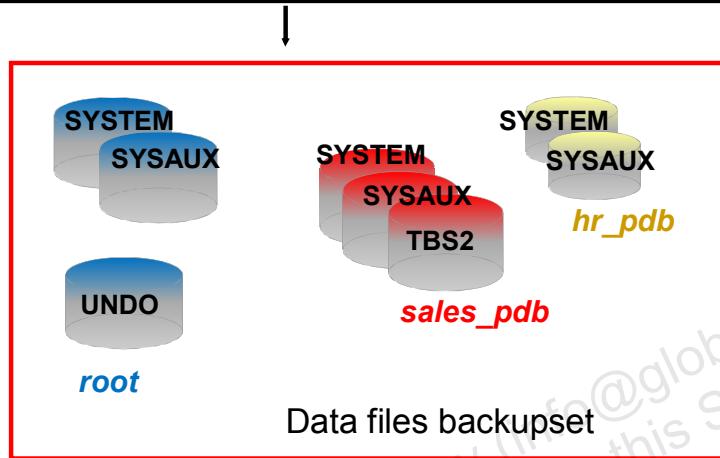
You can also create a backup (either a backup set or image copies) of previous image copies of all data files and control files in the CDB by using the following command:

```
RMAN> BACKUP COPY OF DATABASE;
```

CDB Backup: User-Managed Hot CDB Backup

- Perform a user-managed hot CDB backup.

```
SQL> CONNECT / AS SYSDBA
SQL> ALTER DATABASE BEGIN BACKUP;
SQL> !cp datafiles /backup_dir
SQL> ALTER DATABASE END BACKUP;
```



ORACLE

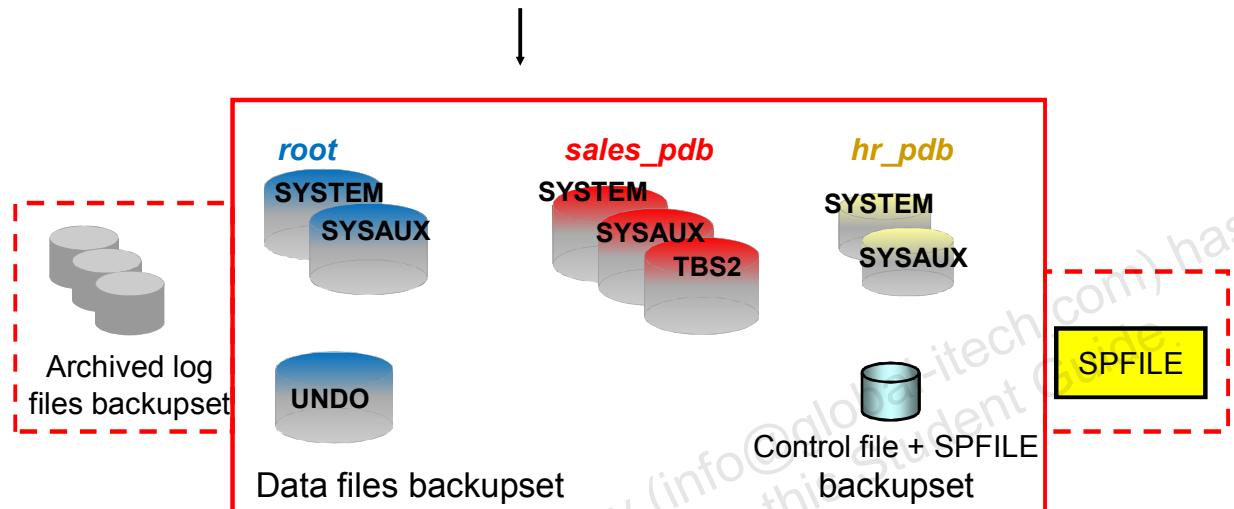
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

After setting the whole CDB in backup mode, all data files except those of the seed PDB are placed in ACTIVE mode. This mode allows you to back up the data files with an O/S command. The data files of the read only seed container can be backed up any time because they are not subject to any write operation.

CDB Backup: Partial CDB Backup

Back up the root and/or individual PDBs.

```
RMAN> CONNECT TARGET /
RMAN> BACKUP PLUGGABLE DATABASE "CDB$ROOT", sales_pdb;
RMAN> BACKUP PLUGGABLE DATABASE hr_pdb PLUS ARCHIVELOG;
```



ORACLE

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

A partial CDB backup backs up the entire set of data files of the root, all data files of defined PDBs, and the control file and SPFILE as it has been configured to be backed up automatically after each backup.

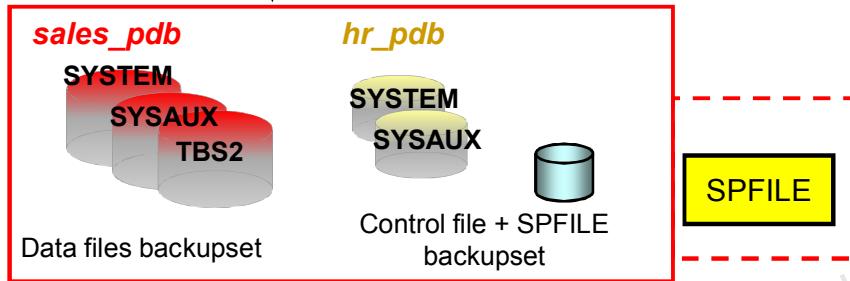
The command `BACKUP PDB "CDB$ROOT" , sales_pdb` backs up all data files of the root container, namely the SYSTEM, SYSAUX, and UNDO data files, and then all data files of `sales_pdb` PDB, namely the SYSTEM, SYSAUX and TBS2 data files.

PDB Backup: Whole PDB Backup

- Backup whole PDBs with RMAN:

```
RMAN> CONNECT TARGET /
RMAN> BACKUP PLUGGABLE DATABASE sales_pdb;
```

```
RMAN> BACKUP PLUGGABLE DATABASE sales_pdb, hr_pdb;
```



- Perform a user-managed hot PDB backup:

```
SQL> ALTER PLUGGABLE DATABASE sales_pdb BEGIN BACKUP;
SQL> !cp pdb_datafiles /backup_dir
SQL> ALTER PLUGGABLE DATABASE sales_pdb END BACKUP;
```

ORACLE

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

A whole PDB backup backs up the entire set of data files of an individual PDB, and the control file and SPFILE as it has been configured to be backed up automatically after each backup.

In the first example, the command `BACKUP PDB` backs up all data files of `sales_pdb` PDB, namely the `SYSTEM`, `SYSAUX` and `TBS2` data files. You could also connect to the PDB target as follows:

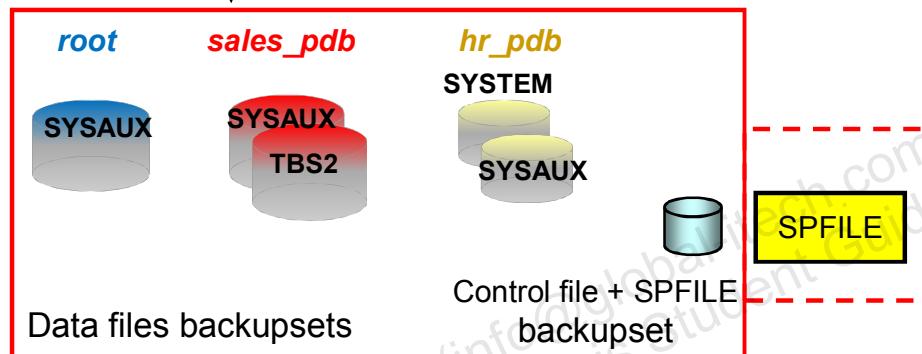
```
$ rman TARGET sys@sales_pdb
```

In the second example, the same command backs up all data files of two named PDBs.

You can also perform user-managed hot backups using SQL*Plus and the new command `ALTER PLUGGABLE DATABASE` with the clause `BEGIN BACKUP` and `END BACKUP` as shown in the third example.

PDB Backup: Partial PDB Backup

```
RMAN> CONNECT TARGET /
RMAN> REPORT SCHEMA;
RMAN> BACKUP TABLESPACE sales_pdb:tbs2;
RMAN> BACKUP TABLESPACE hr_pdb:system,
          sales_pdb:sysaux;
RMAN> BACKUP TABLESPACE sysaux, hr_pdb:sysaux;
```



ORACLE

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

A partial PDB backup backs up the data files named tablespaces of individual PDBs, and the control file and SPFILE as it has been configured to be backed up automatically at each backup performed.

The example uses the command `BACKUP TABLESPACE` to back up all data files of tablespace `TBS2` of `sales_pdb` PDB. To find the names of tablespaces within PDBs, use `REPORT SCHEMA` command.

The second backup uses the same command to back up all data files of tablespace `SYSTEM` of `hr_pdb` PDB and all data files of tablespace `SYSAUX` of `sales_pdb` PDB.

The third backup uses the same command to back up all data files of tablespace `SYSAUX` of the `root` and all data files of tablespace `SYSAUX` of `hr_pdb` PDB.

Recovery

- Instance recovery: **CDB level only**
- Automatic missing temp file recreation at **CDB open**
- Complete media recovery after file loss or corruption
 - **CDB mounted:** same as for non-CDB
 - Redo log files, control files
 - SYSTEM / UNDO root data files
 - **SYSTEM PDB data files**
 - **PDB opened:** Any other PDB data file than SYSTEM
 - **Tablespace OFFLINE:** any other PDB or CDB data file
- Incomplete media recovery after file loss or corruption
 - CDB mounted: The whole CDB back in time
 - PDB closed: A whole PDB back in time
 - TSPITR for **any tablespace** except SYSTEM, UNDO, SYSAUX
- Block recovery: no change
- **Flashback database: CDB mounted**



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

In a CDB, the granularity of media recovery is very flexible and can be done for the entire CDB, for a PDB, for a tablespace, for a data file, or even for a block.

Crash and instance recovery is supported for a CDB as a whole, because there is one single instance for the root and all its PDBs. When the instance crashes, the only possible instance recovery is a CDB instance recovery.

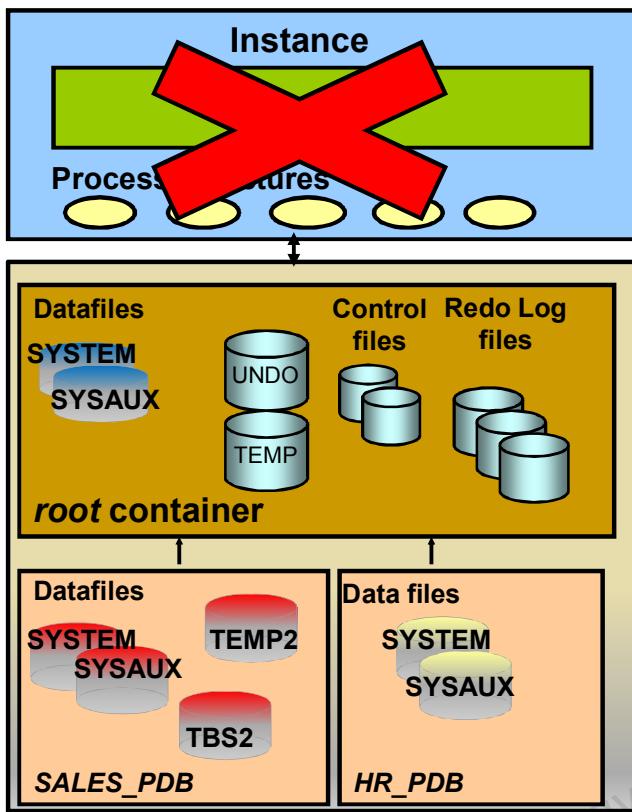
If redo log files or control files, data files from the root SYSTEM or UNDO tablespace are lost, because these files belong to the root, the only possible media recovery is at CDB level.

There exists a new level of media recovery supported, individual PDB media recovery after datafile loss or corruption of a PDB.

There exists a new level of incomplete or PITR recovery, PDB PITR that will look like a tablespace recovery.

Correcting human error applying to the root and PDBs requires flashback at the CDB level only. For example, if you dropped a common or local schema, you must use flashback database applying to all containers, the root, and the PDBs.

Instance Failure



PDB instance recovery is **impossible**.

After instance failure:

- Connect to the root
- Open the root
- Open all PDBs with:
 - Triggers

```
SQL> STARTUP;
```

– SQL statement

```
SQL> STARTUP;
SQL> ALTER PLUGGABLE DATABASE
2          ALL OPEN;
```

ORACLE

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

Crash-instance recovery is supported for a CDB as a whole, because there is one single instance for the root and all its PDBs.

Redo log files required to perform the instance recovery are stored in the unique set of redo log files of the CDB. There is a single shared redo stream for root and all the PDBs used to perform instance recovery. The instance recovery is performed while opening the root.

There is no way to perform a PDB instance recovery.

For an instance to open a data file, the system change number (SCN) contained in the data file's header must match the current SCN that is stored in the control files.

If the numbers do not match, the instance applies redo data from the online redo logs, sequentially "redoing" transactions until the data files are up-to-date. After all data files have been synchronized with the control files, the root is opened and by default all PDBs are still in mount state.

When redo logs are applied, *all* transactions are applied to bring the CDB up to the state as of the time of failure. This usually includes transactions that are in progress but have not yet been committed. After the root has been opened, the uncommitted transactions are rolled back on root data files. After the PDBs are opened, the uncommitted transactions are rolled back on PDBs data files. When opening a CDB, by default all PDBs remain in mounted state. Either triggers are automatically executed to open the PDBs, or you execute the command

`ALTER PLUGGABLE DATABASE ALL OPEN` to open them all.

NOARCHIVELOG Mode

If the database is in NOARCHIVELOG mode, and a data file is lost, perform the following tasks:

- Shut down the instance if it is not already down.
- Restore the entire CDB including all data files and control files.
- Start up the instance and open the CDB and all PDBs.

Users must re-enter all changes made since the last backup.



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

The loss of any data file from a CDB in NOARCHIVELOG mode requires complete restoration of the CDB, including control files and all data files of the root and all PDBs. If you have incremental backups, then you need to perform the restore and recover operations on the CDB.

With the database in NOARCHIVELOG mode, recovery is possible only up to the time of the last backup. So users must reenter all changes made since that backup.

For this type of recovery, proceed with the following steps:

- Shut down the instance if it is not already down.
- Restore the entire CDB including all data and control files from the last backup.
- Start up the instance and open the root and PDBs.

In the next slides, you consider that the CDB is in ARCHIVELOG mode.

Media Failure: CDB or PDB Temp File Recovery

SQL statements that require temporary space to execute may fail if one of the temp files is missing: similar to non-CDB.

```
SQL> CONNECT / AS SYSDBA
SQL> select * from DBA_OBJECTS
      2          order by 1,2,3,4,5,6,7,8,9,10,11,12,13;
select * from DBA_OBJECTS order by
1,2,3,4,5,6,7,8,9,10,11,12,13
      *
ERROR at line 1:
ORA-01565: error in identifying file
'/u01/app/oracle/oradata/CDB1/temp01.dbf'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
```

- Automatic re-creation of temporary files at **CDB opening**
- Manual re-creation also possible



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

When you create a user, you can specify a temporary tablespace to be used by the user. If a temporary tablespace is not specified, the default tablespace for the PDB is used. If a default tablespace is not specified for the PDB, the temporary tablespace for the CDB is used.

If a temp file belonging to the CDB temporary tablespace is lost or damaged, and the user issuing the statement uses it, an error during the execution of SQL statements that require this temporary space occurs.

The SQL statement shown in the slide has a long list of columns to order by, which results in the need for temporary space from the root temporary tablespace. The missing file error is encountered when this statement requiring a sort is executed.

The CDB instance can start up with a missing temporary file. If any of the temporary files do not exist when the CDB instance is started, they are created automatically and the CDB opens normally. When this happens, a message like the following appears in the alert log during startup:

- Re-creating the temp file /u01/app/oracle/oradata/CDB1/temp01.dbf

You can decide a manual recreation instead, while connected to root:

```
SQL> ALTER TABLESPACE temp ADD TEMPFILE
      2  '/u01/app/oracle/oradata/CDB1/temp02.dbf' SIZE 20M;
SQL> ALTER TABLESPACE temp DROP TEMPFILE
      3  '/u01/app/oracle/oradata/CDB1/temp01.dbf';
```

Media Failure: PDB Temp File Recovery

SQL statements that require temporary space to execute may fail if one of the temp files is missing.

```
SQL> CONNECT local_user@HR_PDB
SQL> select * from my_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13;
select * from my_table order by
1,2,3,4,5,6,7,8,9,10,11,12,13
*
ERROR at line 1:
ORA-01565: error in identifying file
'/u01/app/oracle/oradata/CDB1/HR_PDB/temp2_01.dbf'
ORA-27037: unable to obtain file status
Linux Error: 2: No such file or directory
```

- Automatic re-creation of temporary files at **CDB opening only**
- Manual re-creation also possible



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

If a temp file belonging to a PDB temporary tablespace is lost or damaged, and the user issuing the statement uses it, an error during the execution of SQL statements that require that temporary space for sorting occurs.

The SQL statement shown in the slide has a long list of columns to order by, which results in the need for temporary space from the PDB temporary tablespace. The missing file error is encountered when this statement requiring a sort is executed.

The PDB can open with a missing temporary file. If any of the temporary files do not exist when the PDB is opened, they are not created automatically. They are automatically recreated at CDB startup.

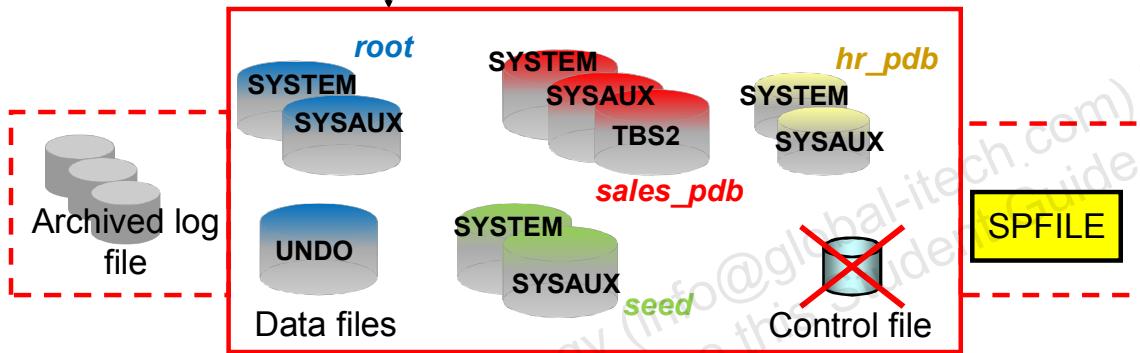
You can perform a manual recreation instead, while connected to the PDB:

```
SQL> ALTER TABLESPACE temp ADD TEMPFILE
  2  '/u01/app/oracle/oradata/CDB1/HR_PDB/temp2_02.dbf'
  3  SIZE 20M;
SQL> ALTER TABLESPACE temp DROP TEMPFILE
  2  '/u01/app/oracle/oradata/CDB1/HR_PDB/temp2_01.dbf';
```

Media Failure: Control File Loss

Similar to non-CDBs: CDB mounted

```
RMAN> CONNECT TARGET /
RMAN> STARTUP NOMOUNT;
RMAN> RESTORE CONTROLFILE FROM AUTOBACKUP;
RMAN> ALTER DATABASE MOUNT;
RMAN> RECOVER DATABASE;
RMAN> ALTER DATABASE OPEN RESETLOGS;
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;
```



ORACLE

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

If a control file is missing or corrupted, because control files belong to the CDB, the instance soon crashes and a whole CDB media recovery is required.

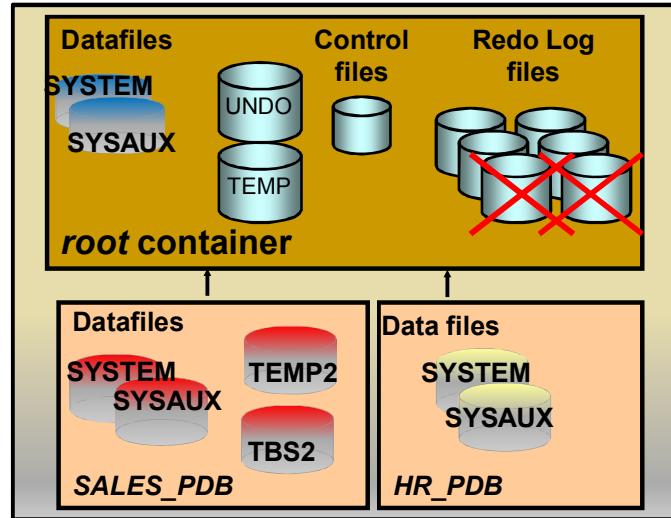
1. First start the CDB instance.
2. Then restore the control file from a backup.
3. Mount the CDB.
4. Then recover and open the CDB in resetlogs.

Two possibilities:

```
RMAN> ALTER DATABASE OPEN RESETLOGS;
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;
Or
SQL> ALTER DATABASE OPEN RESETLOGS;
```

Note: With SQL*Plus, the trigger AFTER STARTUP ON DATABASE to open the PDBs is fired. This is not the case with RMAN.

Media Failure: Redo Log File Loss



Similar to non-CDB procedures:

1. Connect to the root container.
2. Examine the STATUS of the lost file: ACTIVE, INACTIVE, CURRENT.
3. Proceed as with non-CDBs.

ORACLE

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

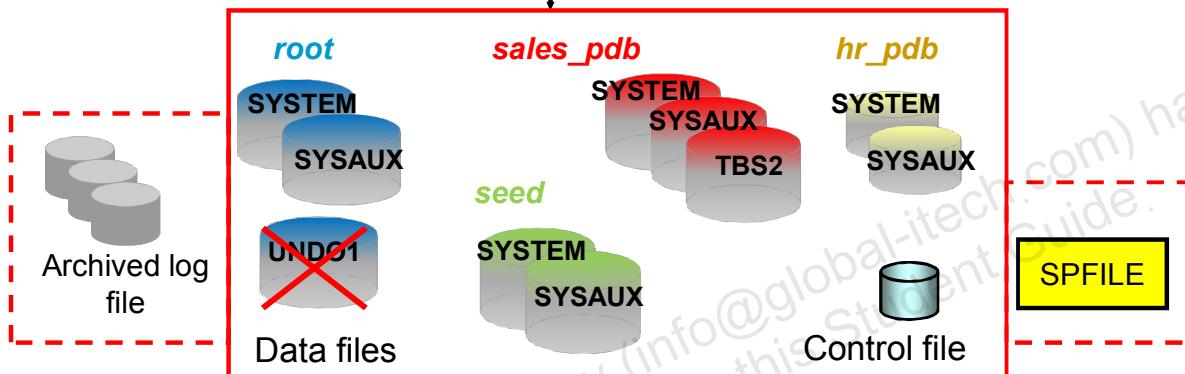
Missing or corrupted redo log files, because there is only one redo stream for the CDB instance (or one redo stream for each instance of a RAC CDB), require a whole CDB media recovery as explained in the slide.

Depending on whether a whole redo log group or only a redo log member is missing, follow the same procedures as those for non-CDBs.

Media Failure: Root SYSTEM or UNDO Data File

Similar to non-CDBs: CDB mounted

```
RMAN> STARTUP MOUNT;
RMAN> RESTORE TABLESPACE und01;
RMAN> RECOVER TABLESPACE und01;
RMAN> ALTER DATABASE OPEN;
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;
```



ORACLE

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

If the missing or corrupted data file belongs to the root container SYSTEM or UNDO tablespace, then the CDB instance will require shutdown, and a media recovery is required. In a RAC environment, you would shut down all instances of the CDB.

This means that all PDBs will be closed.

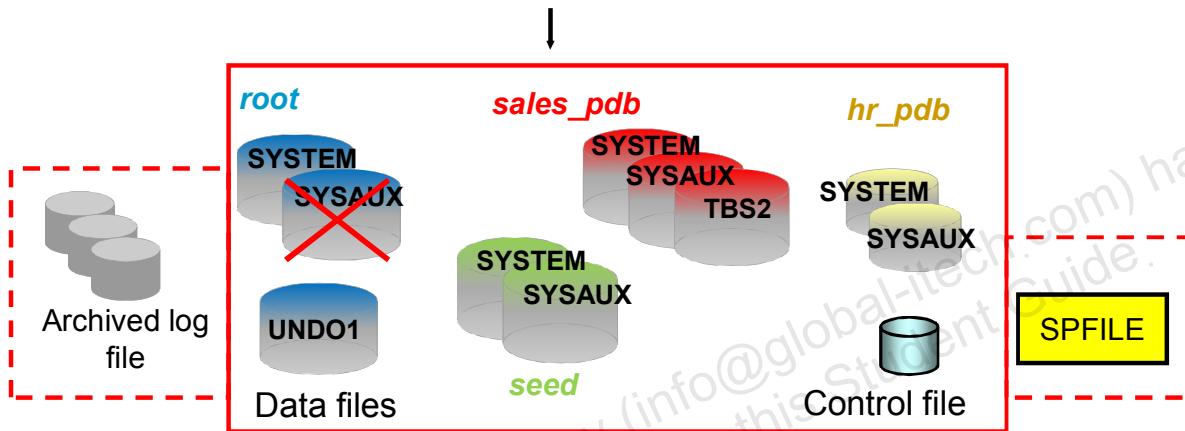
The CDB must be mounted before restoring and recovering the missing root data file.

After the root data file is recovered, open the CDB and all PDBs.

Media Failure: Root SYSAUX Data File

Similar to non-CDBs: tablespace OFFLINE

```
RMAN> ALTER TABLESPACE sysaux OFFLINE IMMEDIATE;
RMAN> RESTORE TABLESPACE sysaux;
RMAN> RECOVER TABLESPACE sysaux;
RMAN> ALTER TABLESPACE sysaux ONLINE;
```



ORACLE

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

If the data file missing or corrupted is a data file of the root tablespaces other than SYSTEM or UNDO, you offline the tablespace, and then perform a tablespace media recovery.

This means that no PDB is closed.

The CDB remains opened while restoring and recovering the missing root data file.

After the root data file is recovered, bring the tablespace online.

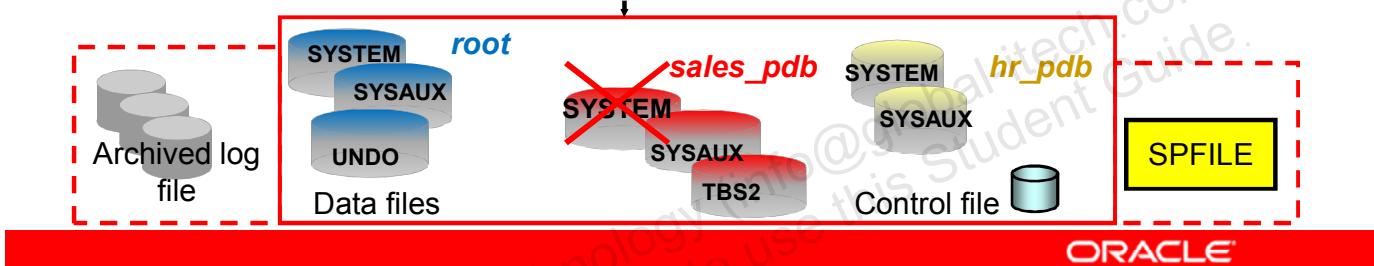
Media Failure: PDB SYSTEM Data File

- If the PDB is opened, the CDB can only be mounted.
- If the PDB is closed, the CDB can be left opened.
 - Restore and recover the PDB:

```
RMAN> STARTUP MOUNT;          - useless on closed PDB -
RMAN> RESTORE PLUGGABLE DATABASE sales_pdb;
RMAN> RECOVER PLUGGABLE DATABASE sales_pdb;
RMAN> ALTER DATABASE OPEN;    - useless on closed PDB -
RMAN> ALTER PLUGGABLE DATABASE sales_pdb OPEN;
```

- Restore and recover the missing tablespace or data file:

```
RMAN> RESTORE TABLESPACE sales_pdb:system;
RMAN> RECOVER TABLESPACE sales_pdb:system;
```



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

If the data file missing or corrupted belongs to a PDB and more specifically to the SYSTEM tablespace, the CDB must be closed unless the PDB is already closed.

A pluggable database or tablespace or data file media recovery is required before the PDB can be reopened.

If the PDB was closed at the time issue, the users can still work in other PDBs during the PDB recovery.

If the PDB was still opened at the time issue, users cannot work at all in any other PDB because the CDB needs to be shut down and mounted only.

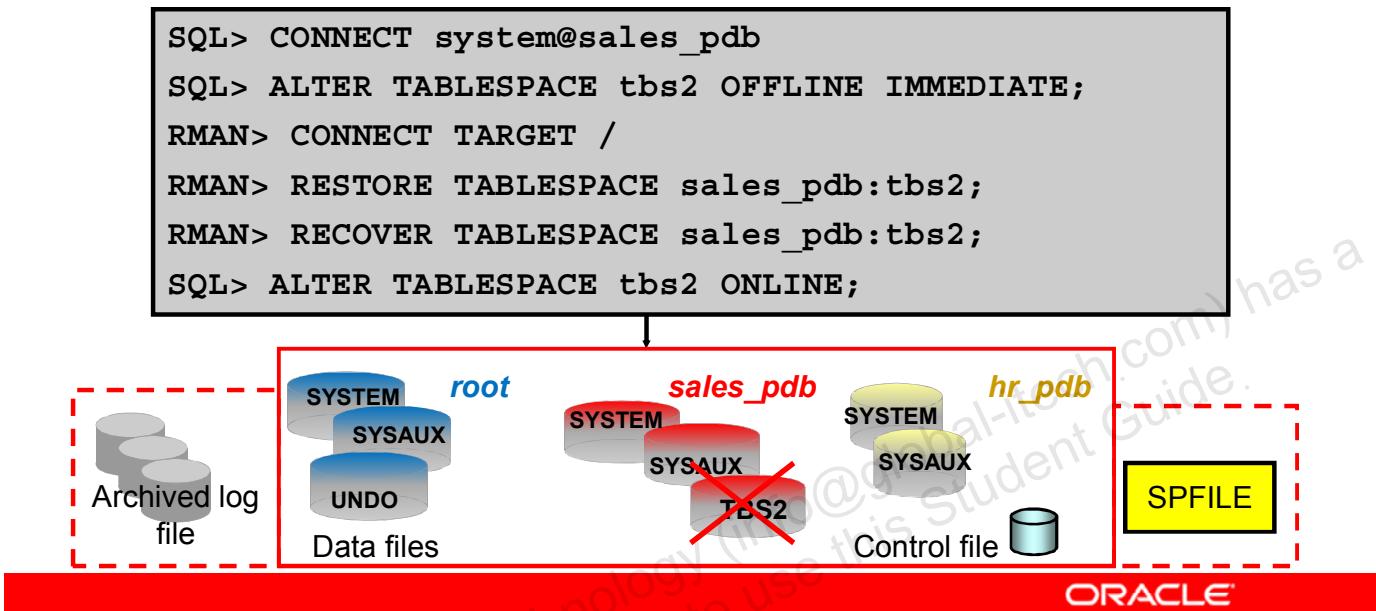
The recovery must be issued from root.

Media Failure: PDB Non-SYSTEM Data File

Similar to non-CDBs: perform the recovery within the PDB

- Connect to the PDB.
- Put the tablespace OFFLINE.
- Other PDBs are not impacted.

```
SQL> CONNECT system@sales_pdb
SQL> ALTER TABLESPACE tbs2 OFFLINE IMMEDIATE;
RMAN> CONNECT TARGET /
RMAN> RESTORE TABLESPACE sales_pdb:tbs2;
RMAN> RECOVER TABLESPACE sales_pdb:tbs2;
SQL> ALTER TABLESPACE tbs2 ONLINE;
```



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

If the data file missing or corrupted belongs to a PDB and more specifically to any tablespace other than SYSTEM tablespace, the PDB need not be closed. The data file with the error is taken OFFLINE IMMEDIATE. Media recovery for that data file is required and performed in a similar way to media recovery of a set of tablespaces. During that recovery time, users can work with other PDB tablespaces and within other PDBs.

The recovery must be issued from the root.

Media Failure: PITR

- PDB PITR

```
RMAN> ALTER PLUGGABLE DATABASE PDB1 CLOSE;
RMAN> RUN {
      SET UNTIL SCN = 1851648 ;
      RESTORE pluggable DATABASE pdb1;
      RECOVER pluggable DATABASE pdb1
          AUXILIARY DESTINATION='/u01/app/oracle/oradata';
      ALTER PLUGGABLE DATABASE pdb1 OPEN RESETLOGS;
    }
```

- PDB Tablespace PITR

```
RMAN> RECOVER TABLESPACE PDB1:TEST_TBS
      2>           UNTIL SCN 832972
      3>           AUXILIARY DESTINATION '/tmp/CDB1/reco';
RMAN> ALTER TABLESPACE PDB1:TEST_TBS ONLINE;
```



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

If you need to recover a PDB database to a point in time in the past beyond flashback retention, then in this case, flashback is not possible, therefore a point-in-time recovery is necessary.

Recovering a PDB to a point-in-time does not affect all parts of the CDB: the whole CDB is still opened and therefore all other PDBs are opened. After recovering a PDB to a specified point-in-time, when you open the PDB using the `RESETLOGS` option, a new incarnation of the PDB is created. The PDB `RESETLOGS` does not perform a `RESETLOGS` for the CDB.

- A PDB record in the control file is updated.
- Each redo log record carries PDB id in the redo header. This is how recovery knows which redo applies to which PDB. Redo logs are shared by all PDBs; redo from each PDB is written to a single set of redo logs.

Conceptually a PDB resetlogs is similar to a database resetlogs.

After recovery, the old backup of the PDB remains valid and can be used if a media failure occurs. After restoring/recoverring a PDB to a past point in time, one cannot open the PDB read only. PDB read-write open through resetlogs is required.

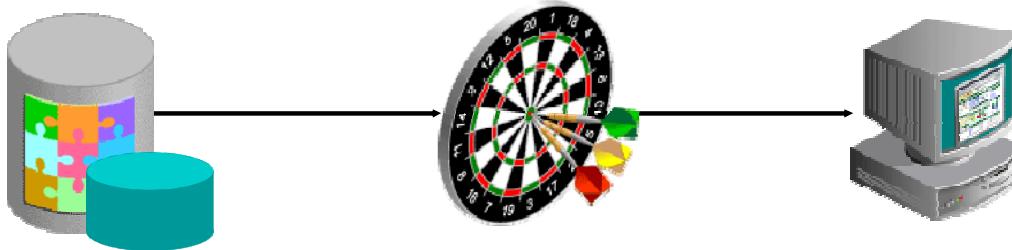
A PDB incarnation is a subincarnation of the CDB. For example, if the CDB is incarnation 5, and a PDB is incarnation 3, then the fully specified incarnation number of the PDB is (5, 3). The initial incarnation of a PDB is 0. To view the incarnation of a PDB, query the `V$PDB_INCARNATION` view.

If you do not use a fast recovery area, you must specify the temporary location of the auxiliary set files by using the `AUXILIARY DESTINATION` clause (example in the slide).

Each PDB has its own set of tablespaces. TSPITR can be used to recover a tablespace to an earlier point in time. Perform the TSPITR as described in the second example on the slide, specifying the full tablespace name including the PDB name.

Recovering a tablespace of a PDB to a point-in-time does not affect all parts of the CDB: the whole CDB is still opened and therefore all PDBs are opened. After recovering a tablespace back in time, put the tablespace back online.

Flashback CDB



1. Configure the FRA.
2. Set the retention target.
3. Enable Flashback Database.

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP MOUNT
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER SYSTEM SET
  2  DB_FLASHBACK_RETENTION_TARGET=2880 SCOPE=BOTH;
SQL> ALTER DATABASE FLASHBACK ON;
SQL> ALTER DATABASE OPEN;
```

If the CDB is in ARCHIVELOG mode, there is no need to restart it.

- No flashback of root without flashing back the whole CDB
- No flashback to a point earlier than the point at which PDBPITR was performed

ORACLE

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

You can configure Flashback Database for the CDB as you would do for any non-CDB:

1. Configure the Fast Recovery Area.
2. Set the retention target with the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter.
3. Enable Flashback Database with the following command:

```
SQL> ALTER DATABASE FLASHBACK ON;
```

Before you can issue the command to enable Flashback Database, the database must be configured for archiving.

You can disable Flashback Database with the `ALTER DATABASE FLASHBACK OFF` command. As a result, all existing Flashback Database logs are deleted automatically.

Restrictions:

- You cannot flash back the root alone without flashing back the entire CDB.
- Flashback Database operations on a CDB may not be permitted if point-in-time recovery has been performed on any of its PDBs. When point-in-time recovery is performed on a PDB, you cannot directly rewind the CDB to a point that is earlier than the point at which DBPITR for the PDB was performed.

Flashback CDB

A common user is dropped.

1. Flashback CDB: CDB mounted in exclusive mode

```
SQL> STARTUP MOUNT
SQL> FLASHBACK DATABASE TO SCN 53943;
```

2. To review changes: open CDB and PDBs in READ ONLY

```
SQL> ALTER DATABASE OPEN READ ONLY;
SQL> ALTER PLUGGABLE DATABASE ALL OPEN READ ONLY;
```

3. To finalize: Flash back again if necessary and open CDB with RESETLOGS .

```
RMAN> SHUTDOWN IMMEDIATE
RMAN> STARTUP MOUNT
RMAN> FLASHBACK DATABASE TO SCN 10;
RMAN> ALTER DATABASE OPEN RESETLOGS;
RMAN> ALTER PLUGGABLE DATABASE ALL OPEN;
```

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

A common schema has been accidentally dropped in the root. You have to flash back the CDB to the time before the common user was dropped impacting all PDBs.

You can use the RMAN FLASHBACK DATABASE command to execute the Flashback Database operation. You can use SEQUENCE and THREAD to specify a redo log sequence number and thread as a lower limit.

Alternatively, you can use the SQL FLASHBACK DATABASE command to return the database to a past time or SCN. If you use the TO SCN clause, you must provide a number. If you specify TO TIMESTAMP, you must provide a time stamp value. You can also specify a restore point name.

Note

- The CDB must be mounted in exclusive mode to issue the FLASHBACK DATABASE command and opened read-only to review changes. The CDB must be opened read/write with the RESETLOGS option when finished.
- When the CDB is opened in READ ONLY mode, the PDBs are still mounted. Open PDBs in READ ONLY mode, too, to review changes.

Special Situations

- Creating a control file backup script:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

```
CREATE CONTROLFILE ...
  datafile sales_pdb file1
  datafile sales_pdb file1
  ...
  datafile hr_pdb file1 ...;
```

- PDB close **incompatible** with CDB hot backup:

```
SQL> ALTER DATABASE BEGIN BACKUP;
SQL> ALTER PLUGGABLE DATABASE hr_pdb CLOSE;
ALTER PLUGGABLE DATABASE pdb1_1 close
*
ERROR at line 1:
ORA-01149: cannot shutdown - file 10 has online backup set
ORA-01110: data file 10: '/D1/oradata/cdb1/pdb1_1/users01.dbf'
```

ORACLE

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

- Besides the control file backup with RMAN auto-backup configuration, you can also use the SQL statement that generates a script including the CREATE CONTROLFILE statement with all data files of the CDB, including the root and PDBs.
The statement is still the same as for non-CDBs:

```
SQL> ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```
- While performing a hot CDB backup, you cannot close any PDB.

Data Dictionary Views: RC_PDBS

New views

- RC_PDBS

New column PLUGGABLE_DBID in V\$ views and RC_xxx views

- V\$DATAFILE_COPY
- V\$BACKUP_DATAFILE
- V\$PROXY_DATAFILE
- RC_DATAFILE_COPY
- RC_BACKUP_DATAFILE
- RC_PROXY_DATAFILE



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

There is a new recovery catalog view RC_PDBS:

- PDB_KEY is the primary key for this PDB in the recovery catalog.
- DB_KEY is the primary key for the CDB in the recovery catalog. Use this column to join with almost any other catalog view.
- NAME is the name of the PDB.
- CON_ID is the unique identifier of the CDB.
- DBID is the unique identifier of the PDB.
- DROP_CHANGE# is the SCN at which the PDB was unplugged from the CDB.
- DROP_TIME is the time at which the PDB was unplugged from the CDB.

The principal V\$ and recovery catalog views contain a new column PLUGGABLE_DBID referencing the unique ID of a PDB, information stored in control file and data file headers.

Quiz

Which statements are true about PDB and CDB backups?

- a. You can connect to a specific target PDB in RMAN.
- b. You can back up a whole PDB only if connected to the target PDB in RMAN.
- c. You may back up specific PDB tablespaces with a tablespace backup.
- d. You may back up the root container with a container - specific backup.



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

Answer: a, c, d

Quiz

You can recover a single PDB, even if the CDB is in mounted state:

- 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:

- Perform CDB and PDB backups
- Recover CDB from essential file loss
- Recover PDB from PDB data file loss
- Perform flashback database

Practice 6 Overview: Managing CDB and PDBs Backup and Recovery

These practices cover the following topics:

- Cold backup of a CDB
- Whole CDB backup with RMAN
- PDB backup with RMAN
- Recovery from SYSTEM PDB data file loss
- Recovery from non-essential PDB data file loss

Optional practices:

- SQL PDB Hot backup
- SQL control file backup
- Recovery from all control files loss
- Recovery from redo log member loss
- Recovery from SYSTEM root data file loss
- Recovery from a non-essential root data file loss
- Point-in-time recovery of PDB tablespaces
- CDB flashback from common user DROP



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

Automatic Data Optimization and Storage Enhancements



ORACLE®

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

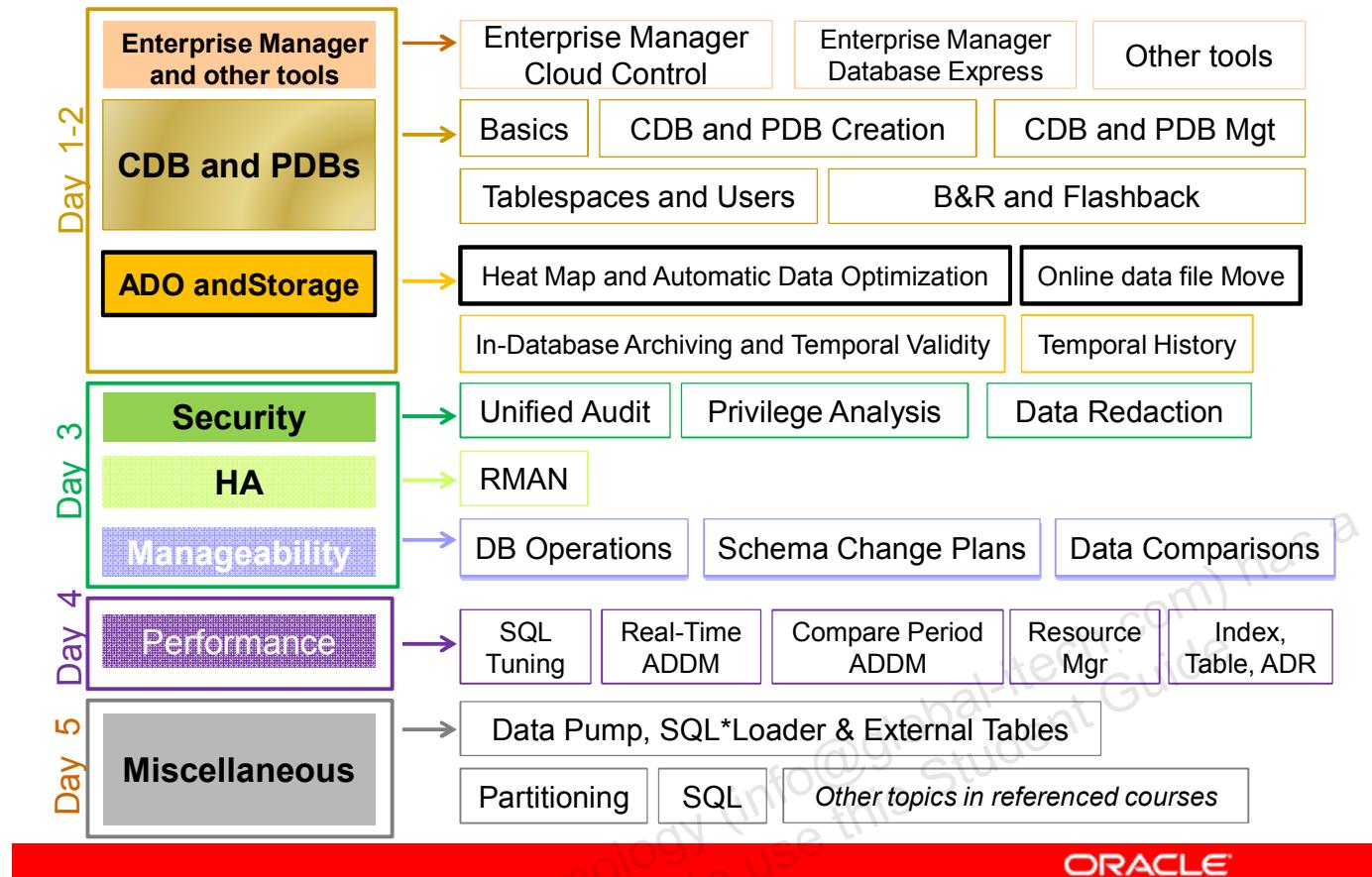
Heat Map, Automatic Data Optimization and Online Data File and Partition Move



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.

Heat Map and Automatic Data Optimization: Heat Map and Automatic Data Optimization (ADO) can be used to implement your ILM (Information Lifecycle Management) strategy, along with Partitioning, Advanced Compression, and Hybrid Columnar Compression.

Storage: The Oracle Database 12c **Online Move data file** feature provides the capability to move an online data file from one kind of storage system to another while the database is opened and accessing the file. The Oracle Database 12c **Online Move partition** feature provides the capability to move and compress a partition while the partition is being accessed by DML statements.

Objectives

After completing this lesson, you should be able to:

- Describe activity tracking with Heat Map
- Use views to display Heat Map statistics
- Describe Automatic Data Optimization
- Create ADO policies
- Use views and procedures to monitor Automatic Data Optimization
- Move a data file online
- Move a partition online



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

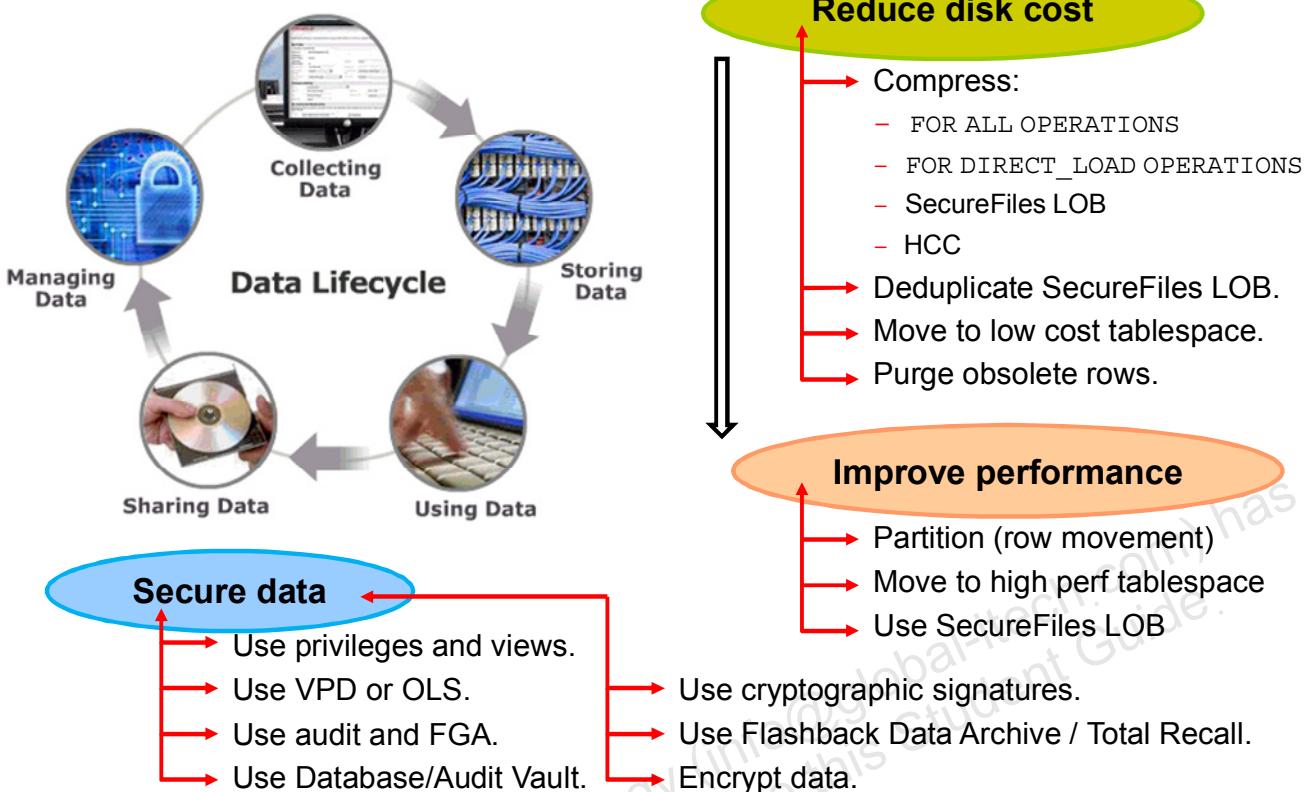
Note: For a complete understanding of Heat Map and Automatic Data Optimization new features and usage, and Online Move data file and partitions features, refer to the following guides in the Oracle documentation:

- *Oracle Database VLDB and Partitioning Guide 12c Release 1 (12.1)*
- *Oracle Database Administrator's Guide 12c Release 1 (12.1)*
- *Oracle Database PL/SQL Packages and Types Reference 12c Release 1 (12.1) – “DBMS_ILM”, “DBMS_ILM_ADMIN” , and “DBMS_HEAT_MAP” chapters*

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

- *Oracle Database 12c New Features Demo Series* demonstrations:
 - *Online data file Move*
- *Oracle By Example (OBE)*:
 - *Setting Up Compression Tiering for Automatic Data Optimization*
 - *Setting Up Storage Tiering Automatic Data Optimization*

ILM Challenges and Solutions



ORACLE

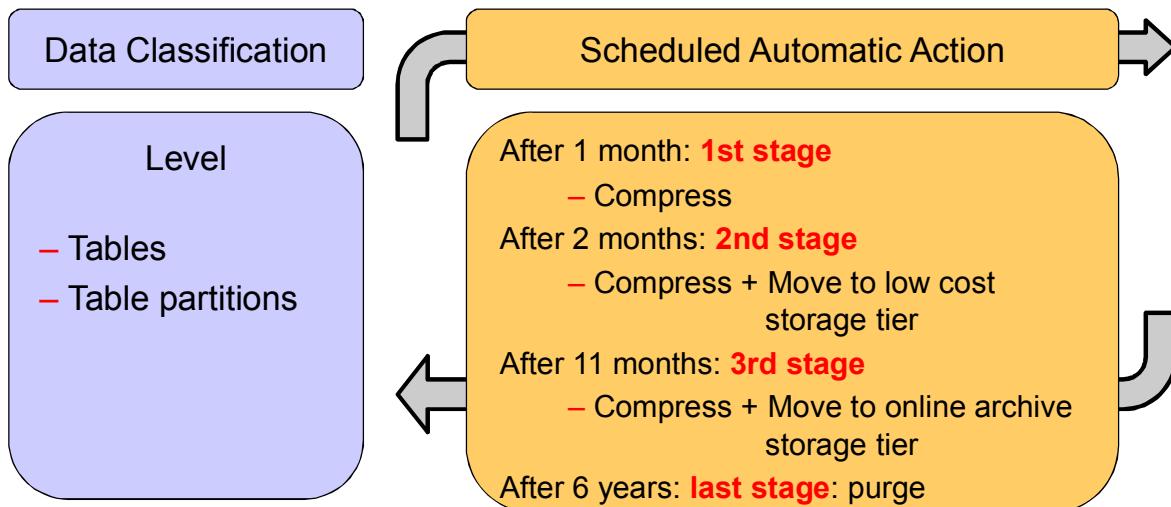
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

What is Information Lifecycle Management, also referred to as ILM? It is a strategy for managing business data over its lifetime in order to reduce storage costs, improve data access within the database and adapt to regulatory requirements. ILM is the practice of applying policies for the effective management of information throughout its useful life. ILM includes every phase of a “row.” It consists of the policies, processes, practices, and tools that are used to align the business value of information with the most appropriate and cost-effective IT infrastructure from the time information is conceived through its final disposition.

Customers of Oracle Database are interested in leveraging compression and storage tiering for satisfying their ILM needs. While for many customers compression may be sufficient, a subset of customers wants to optimize dormant data further by moving it to high-density, low-cost Oracle Storage or to other lower-cost storage.

In order to utilize storage tiering or compression tiering for dormant data (say, sales orders more than one year old), online transaction processing (OLTP) applications need to implement Partitioning for the larger tables. Oracle supports Partitioning and provides high-level documentation on how to use it, but it is left to the customer to implement and validate Partitioning.

ILM Components



Lifecycle definition with ILM Assistant:

Stage:	Current Orders	Previous Orders	Old Orders	Very Old Orders	End of Life for Orders
Tier:	High Performance	High Performance	Low Cost	Online Archive	
Attributes:	None	Compress	Compress	Compress	
Retention:	Current Month	2 Months	11 Months	6 Years	Purged
Time:	Most Recent Data →→→ Oldest Data				

ORACLE

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

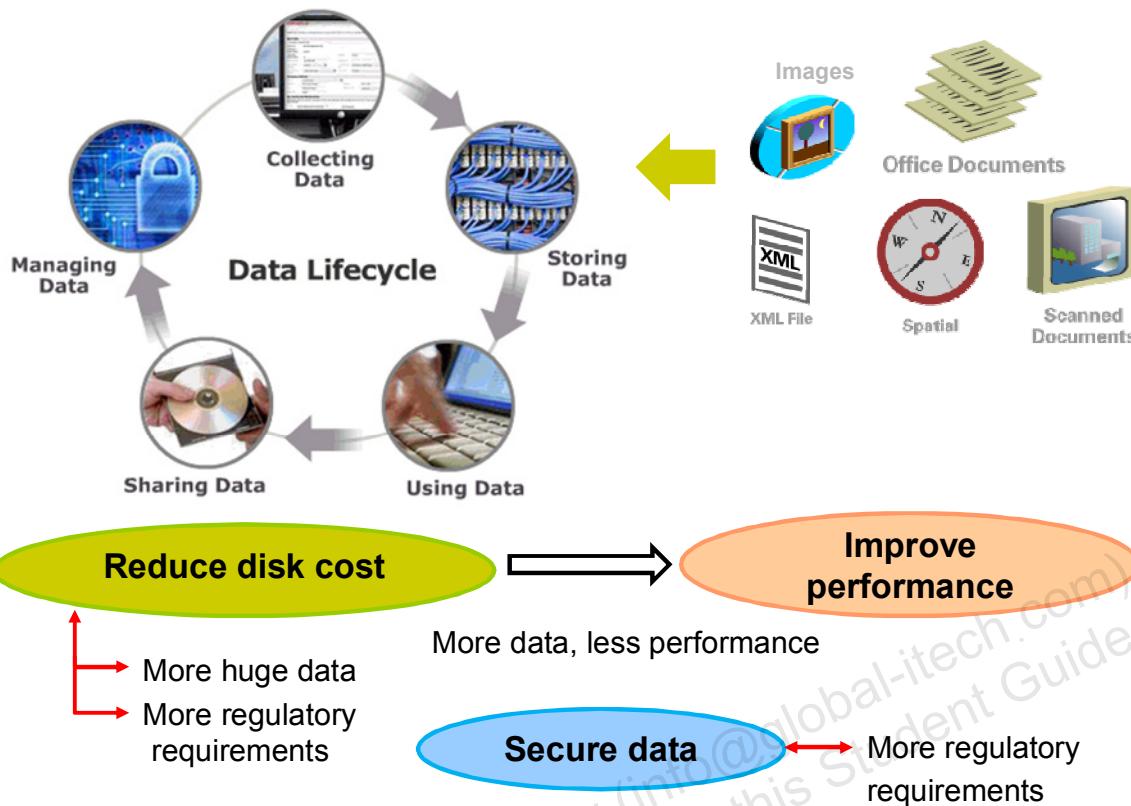
How does a lifecycle definition work in Oracle Database 11g ILM? Consider a typical ILM flow implemented by a major enterprise: data is first bulk-loaded into a table. At this stage, the database administrator (DBA) compresses only active partitions, including the most active one: OLTP Table Compression is optimized for data that is being concurrently modified.

After a month, activity has subsided, although significant OLTP transactions are still carried out. At this stage, data in the former most active partition stays in OLTP Table compressed format, and a new partition is automatically created.

Two months down the line, the data is rarely modified and accessed rarely as well. At this stage, the partition can be moved to a lower-cost storage tier and also to a higher compression level, with Hybrid Columnar Compression (HCC) on Exadata or Oracle Storage.

After 11 months, data is considered dormant because it is no longer accessed or updated, and is moved to another lower cost storage tier, and the highest compression level with HCC, and marked as read-only data.

ILM Challenges



ORACLE

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

Challenges for ILM in Oracle Database 12c are still the same as in Oracle Database 11g:

- Reduce storage costs
- Improve performance
- Secure data

In addition, the amount of data continues to grow, with unstructured data consuming huge volumes of bytes. A number of companies are compelled to retain data for extended periods of time to meet compliance requirements. As a result, more data is being kept in online mode for longer periods of time to keep it available, which increases storage costs. Managing data growth continues to be *the top priority* for database users. Performance of applications often decreases as a result of data growth.

In Oracle Database 11g, life cycle event scanning and subsequent actions need to be performed manually. In Oracle Database 12c, new solutions allow the user to set policies that define application-specified rules on classified data. The rules enforce data flows automatically, with minimal manual intervention.

Solutions

Reduce disk cost:

- Manage storage more efficiently with different tiers of storage
- Use compression levels.

Improve Performance:

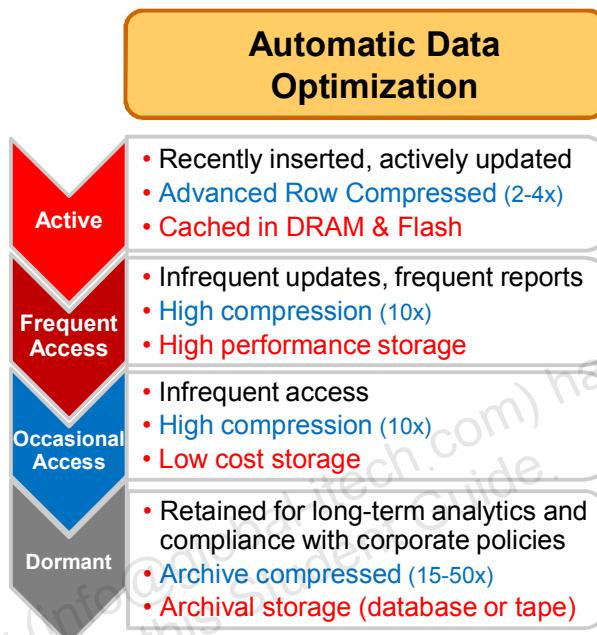
- Compress at different levels

Automatic Data Optimization:

- Policies automatically
 - Compress according to user-defined rules
 - Move data due to space pressure

In-Database Archiving:

- Row-archival
- Temporal Validity



ORACLE

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

Disk Cost and Performance Decrease

As data grows, the typical reaction is to buy and install more disk storage. A smarter approach would be to implement a tiered storage strategy, and put information life cycle strategies in place to manage data better and more cost effectively.

The business requirements are not the same for all data in the database. Data goes through various stages in its life cycle. It starts out as active data, when the data is queried and modified heavily. After a period of time, this data becomes less active. It is a time period when it is queried often (for example, for report generation), but it is rarely modified. In the third phase, the data becomes more or less dormant. It is neither queried nor updated, but it needs to be kept for compliance and regulatory purposes. So it goes to an archiving storage that can still be within the database, and not necessarily outside the database.

It is often necessary to move segments from one tablespace to another. For example, segments can be moved to another tablespace due to space pressure on the tablespace they reside in, thus achieving storage tiering. The ILM infrastructure must manage storage tiering for table segments.

It is also often necessary to move a segment to a new compression level or to move a group of rows from one compression level to another without moving them to a different segment. For example, rows that are no longer likely to be modified can be compressed to row store advanced compression level within the same segment.

What is Heat Map and Automatic Data Optimization (ADO)?

Oracle Database 12c includes activity tracking with Heat Map providing the ability to track and mark data as it goes through life cycle changes:

- Data accesses at segment-level
- Data modifications at block and segment-level

Block-level and segment-level statistics collected in memory are stored in tables in SYSAUX tablespace.

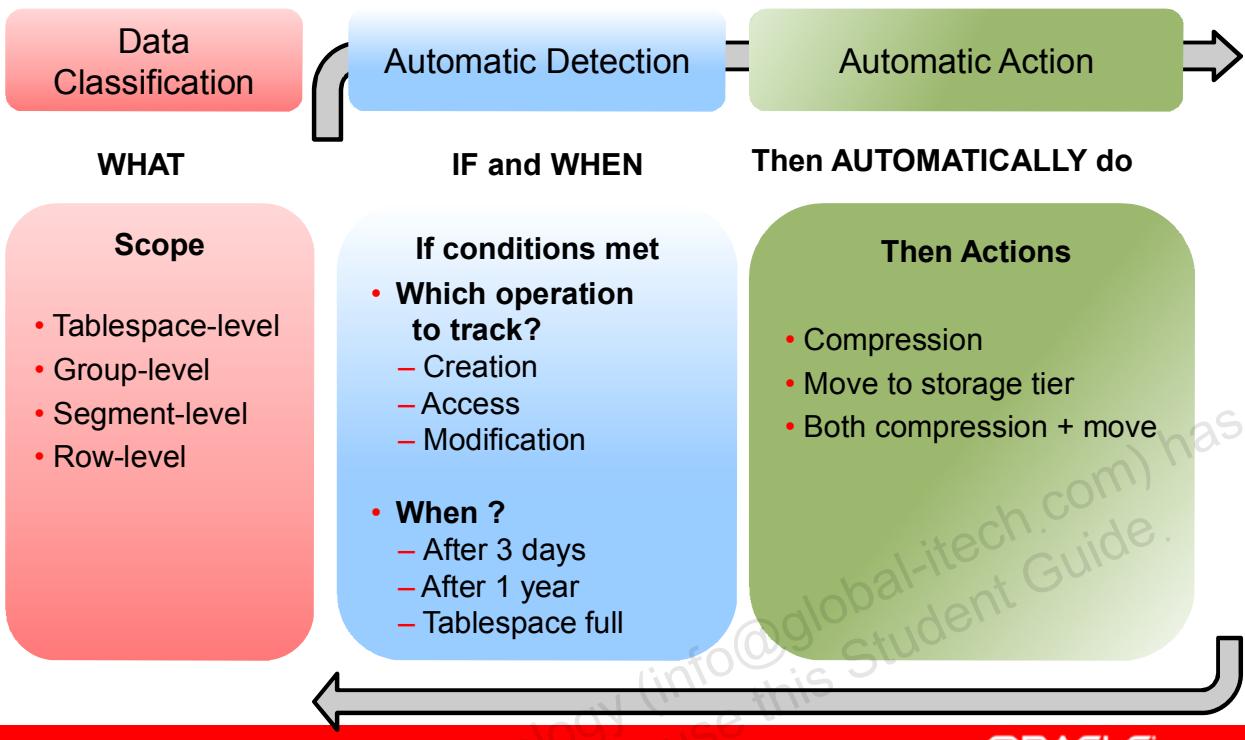
ADO allows you to create policies that use Heat Map statistics to compress and move data only when necessary. ADO automatically evaluates and executes policies that perform compression and storage tiering actions.

How is In-Database Archiving Enhanced in Oracle Database 12c?

- In-Database Archiving solves problems regarding archived and active data. It gives users the ability to keep both active operational data and dormant archived data in the same database tables. While achieving operational efficiency of archiving, applications can easily access only data that is in an operationally active state although archived data is kept in the same table.
- Another new capability, Temporal Validity, creates a valid time dimension for each row of a table, and is another way of indicating operational relevance. Each valid time dimension consists of two date-time columns specified in the table definition. If the data is no longer valid, or not yet valid, this data can easily be hidden from regular queries.

Components

Heat Map and Automatic Data Optimization



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

Consider the same flow as defined previously: data is first bulk-loaded into a table. At this stage, it might be required to have no compression to meet storage and performance service level agreements (SLAs). Then the data is heavily updated and modified for some time, so it is best to leave the data uncompressed. The question is how long this period will last until the number of access and modifications would decrease and you should think about compression and even data movement to another storage tier. How long will the next period last until there will be no more data access and you should think about higher compression and even data movement to an archive storage tier?

- ADO provide the ability to declare policies on different scope levels in the database: tablespace, object, and row-levels, automatically enforced by the ADO infrastructure. An ADO policy is a procedure that executes the registered operations for the requested object to get them to the required state transparently without any user intervention.
- Policies automate data movement between different tiers of storage within the database. They also compress at different levels for each tier, and mainly control when the data movement takes place.

For example, when partition rows were not updated for three days, automatic compression based on policy is performed. This capability obviates the need for the application to make explicit schema changes.

ADO policies allow you to specify the conditions under which row, partition, table access or modification need to be satisfied before a compression or a move action should be performed.

When ADO policies are set at tablespace level as a default behavior, the table, partition, and subpartitions, including new interval partitions stored on that tablespace, are automatically compressed and moved to a lower-cost tablespace based on the policies.

ADO allows the user to set policies that enforce data flows automatically, with minimal intervention.

What Is Automatic Data Optimization?

- ADO policies automatically compress data when it qualifies
- ADO policies automatically move segments when necessary
- ADO is dependent on Heat Map, and will not work unless Heat Map is enabled

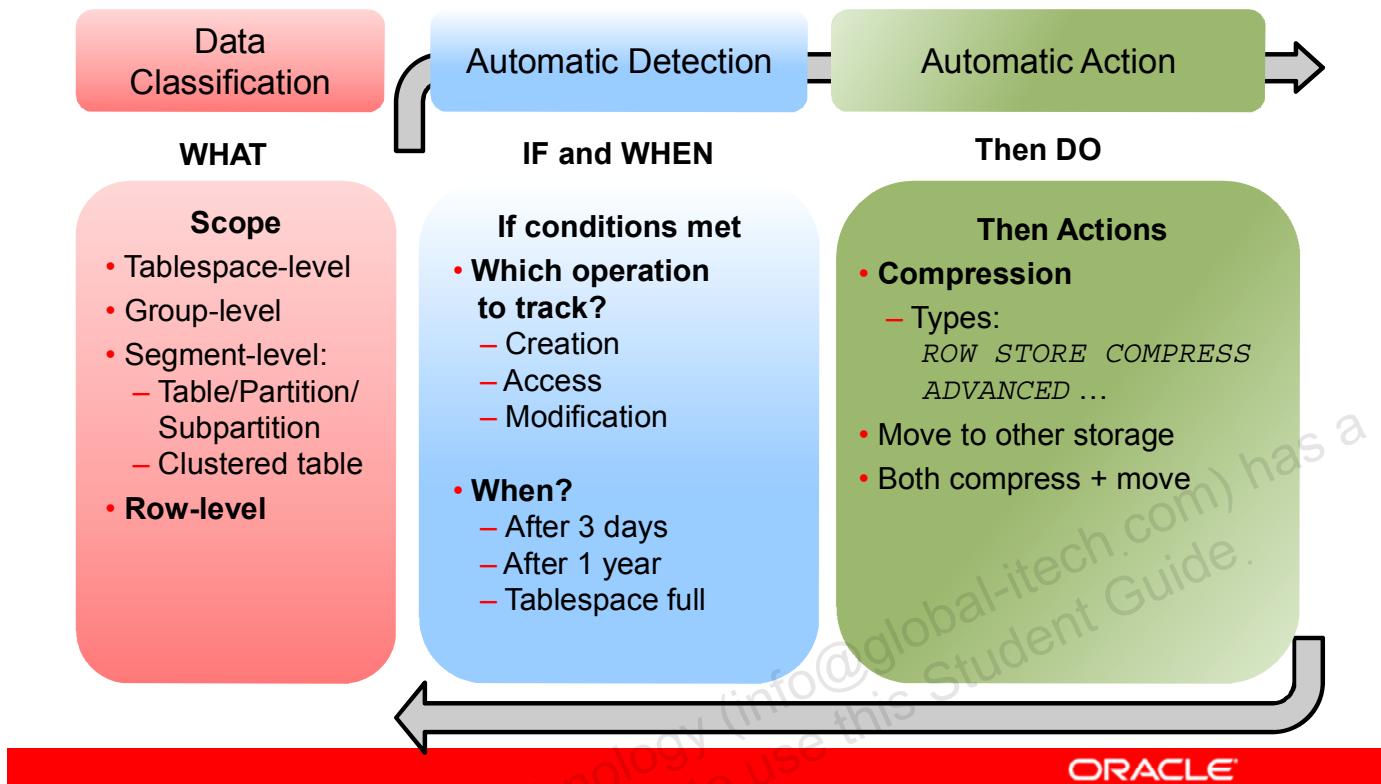
ADO policies automatically execute actions under predefined conditions:

- Compress data only when it qualifies
- Move segments to other storage tiers under space pressure.

ADO can execute compression and data movement only if Heat Map is enabled. After being enabled, Heat Map automatically collects statistics to execute ADO actions after the statistics evaluation is completed.

Data Classification Levels

Activity Tracking and Automatic Data Optimization



ORACLE

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

In Oracle Database 11g, the lowest level to determine data classes is the segment, table, or partition.

In Oracle Database 12c, you can classify each row of a table based on its operational relevance by using the new row-level data classification.

- Modifications tracked at row-level

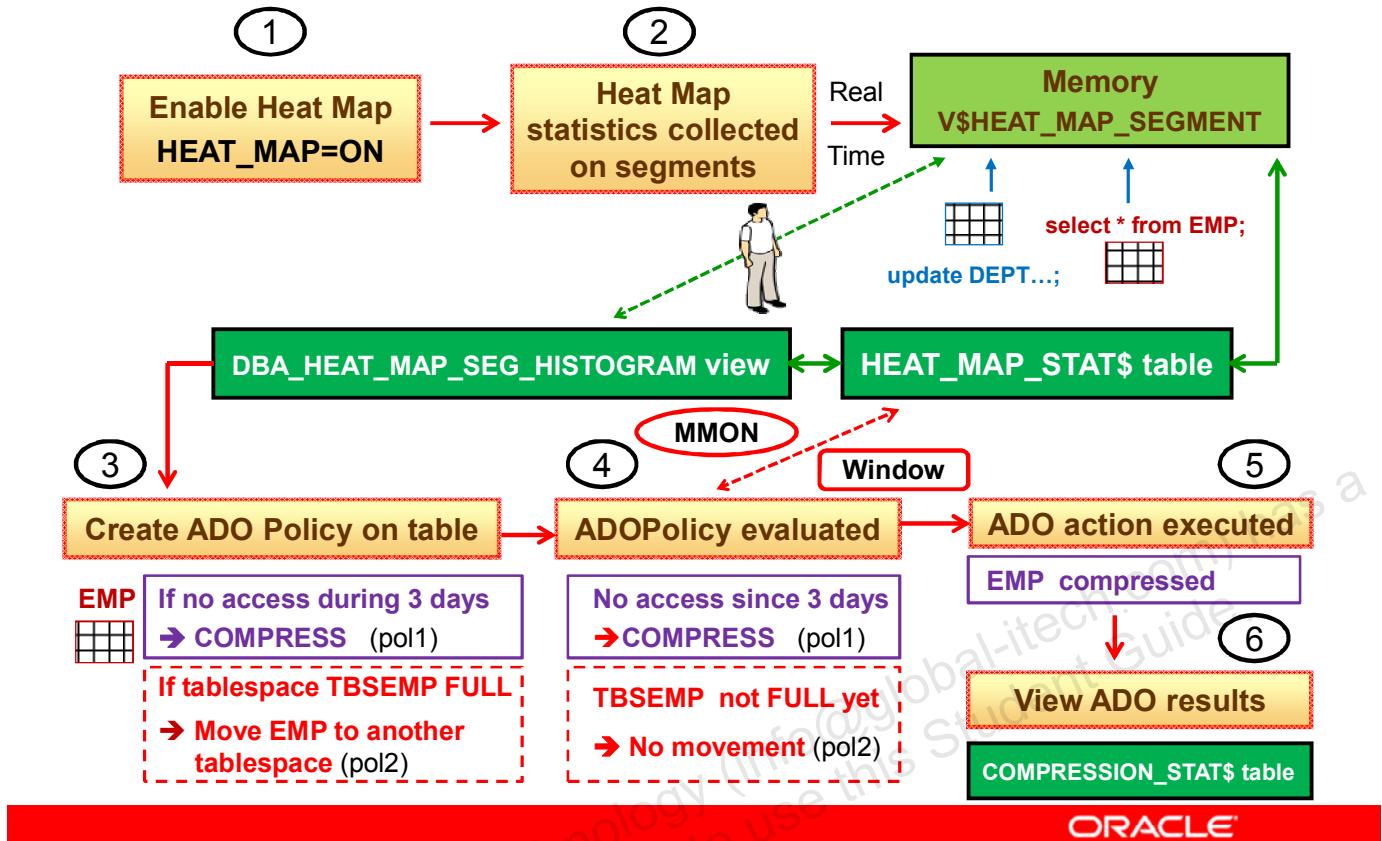
Row-level statistics are managed in-line as part of the work done when a process modifies data.

At segment-level, access and modifications are tracked:

- Last access time:
 - Fetch-by-rowid tracking: collect ROWIDs on access path of index range scan/fetch by ROWID.
 - Full-table-scan tracking
- Last modification time

Segment-level activity tracking information is automatically flushed for persistence by scheduled DBMS_SCHEDULER jobs hourly. This means that segment-level statistics are guaranteed to checkpoint to on-disk statistics tables.

Heat Map and ADO



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

The slide shows how to set up the different steps between Heat Map and ADO to automate the movement of a segment to another tablespace and or the compression of blocks or a segment depending on certain conditions defined in ADO policies.

- The first operation for the DBA is to enable Heat Map, tracking the activity on blocks and segments. Heat Map activates system-generated statistics collection, such as segment access, row and segment modification. Real-time statistics are collected in memory (`V$HEAT_MAP_SEGMENT` view) and regularly flushed by scheduled `DBMS_SCHEDULER` jobs to the persistent table `HEAT_MAP_STAT$`. The persistent data is visible using `DBA_HEAT_MAP_SEG_HISTOGRAM` view.
- The next operation for the DBA is to create ADO policies on segments or groups of segments or as default ADO behavior on tablespaces.
- The next step for the DBA is to schedule when ADO policy evaluation must happen if the default scheduling does not match the business requirements. ADO policy evaluation relies on Heat Map statistics. MMON evaluates row-level policies periodically and start jobs to compress whichever blocks qualify. Segment-level policies are evaluated and executed only during the maintenance window.
- The DBA can then view ADO execution results using `DBA_ILOMEVALUATIONDETAILS` and `DBA_ILOMRESULTS` views.
- Finally, the DBA can verify if the segment moved and is therefore stored on the tablespace defined in the ADO policy and or if blocks or the segment got compressed viewing `COMPRESSION_STAT$` table.

Enabling Heat Map

1. Enable activity tracking or heat map:

- At instance level: setting a new initialization parameter:

```
SQL> ALTER SYSTEM SET heat_map = ON;
```



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

The first step is to enable activity tracking using a new initialization parameter `HEAT_MAP` in order to activate system-generated statistics collection at the instance level. This parameter is dynamic.

Turning the `HEAT_MAP` parameter ON causes DML and access of all segments, other than the ones in the `SYSTEM` and `SYSAUX` tablespaces, to be tracked in memory and then flushed to on-disk statistics tables in the `SYSAUX` tablespace.

Monitoring Statistics: Segment-Level

View segment-level statistics using views:

- DBA_HEAT_MAP_SEG_HISTOGRAM
- DBA_HEAT_MAP_SEGMENT
- V\$HEAT_MAP_SEGMENT

```
SQL> SELECT object_name, subobject_name, track_time,
2       segment_write WRI, full_scan FTS, lookup_scan LKP
3   FROM DBA_HEAT_MAP_SEG_HISTOGRAM;

OBJECT_NAME      SUBOBJECT_NAME TRACK_TIME WRI   FTS   LKP
-----          -----
INTERVAL_SALES
INTERVAL_SALES
INTERVAL_SALES
I_EMPNO          P1 02-JAN-12  YES YES NO
INTERVAL_SALES
INTERVAL_SALES
INTERVAL_SALES
I_EMPNO          P2 28-MAR-12  NO  YES NO
INTERVAL_SALES
INTERVAL_SALES
INTERVAL_SALES
I_EMPNO          P3 28-MAR-12  NO  NO  YES
INTERVAL_SALES
INTERVAL_SALES
INTERVAL_SALES
I_EMPNO          07-dec-12  YES NO  YES
```



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

After you enabled Heat Map, statistics are collected for different activities, data read, and data write. ADO uses these statistics to trigger actions after evaluation.

You have access to these statistics. The DBA_HEAT_MAP_SEG_HISTOGRAM statistics view contains daily segment heat map activity information about tables, table partitions, indexes and LOB segments. After rows are inserted into a table, updated, and selected, statistics are visible in the DBA_HEAT_MAP_SEG_HISTOGRAM and DBA_HEAT_MAP_SEGMENT views.

The views rely on the persistent SYS.HEAT_MAP_STAT\$ table and V\$HEAT_MAP_SEGMENT view that shows real time segment activity information from memory.

The columns of DBA_HEAT_MAP_SEG_HISTOGRAM view are the following:

- TRACK_TIME column: Displays the system time when the segment access was tracked
- SEGMENT_WRITE column: Indicates whether the segment has write access
- FULL_SCAN: Indicates whether segment has full scan
- LOOKUP_SCAN: Indicates whether the segment has lookup scan

segment-level activity tracking and block level modification time tracking are automatically flushed for persistence by hourly scheduled DBMS_SCHEDULER jobs.

Use the procedure and functions of the DBMS_HEAT_MAP package to read heat map statistics from memory.

DBA_HEAT_MAP_SEGMENT View

The latest segment activity tracking time:

```
SQL> SELECT object_name,
  2       segment_write_time WRITE_T, segment_read_time READ_T,
  3       full_scan FTS_T, lookup_scan LKP_T
  4  FROM DBA_HEAT_MAP_SEGMENT;
```

OBJECT_NAME	WRITE_T	READ_T	FTS_T	LKP_T
EMP			07-dec-12	
T1	07-dec-12		08-dec-12	
EMPLOYEE	07-dec-12		08-dec-12	08-dec-12
I_EMPNO	07-dec-12			08-dec-12



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

Monitoring Statistics: Block Level

View block-level statistics using two new PL/SQL table functions.

After rows are modified:

- View the latest modification time of the blocks using `DBMS_HEAT_MAP.BLOCK_HEAT_MAP` function:

```
SQL> SELECT segment_name, tablespace_name, block_id, writetime
  2  FROM table(dbms_heat_map.block_heat_map
  3                  ('SCOTT','EMPLOYEE',NULL,8,'ASC'));

SEGMENT_ TABLESPACE_NAME      BLOCK_ID WRITETIME
-----  -----
EMPLOYEE LOW_COST_STORE          196 12-DEC-12
EMPLOYEE LOW_COST_STORE          197 12-DEC-12
EMPLOYEE LOW_COST_STORE          198 12-DEC-12
```

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

`DBMS_HEAT_MAP.BLOCK_HEAT_MAP` Table Function

The `DBMS_HEAT_MAP.BLOCK_HEAT_MAP` function returns the block-level heatmap for a table segment. The statistic shows the latest modification time of the block.

The input parameters are the following:

- Owner: Owner of the segment
- Segment_name: Table name of a non partitioned table or (sub)partition of partitioned table. It returns no rows when table name is specified for a partitioned table.
- Partition_name: Defaults to NULL. For a partitioned table, specify the partition or subpartition segment name.
- Sort_columnid: ID of the column to sort the output on. Valid values 1..9.
- Sort_order: Defaults to NULL. Possible values: ASC, DESC

The output parameters are the following:

- Owner: Owner of the segment
- Segment_name: Segment name of the non partitioned table
- Partition_name: Partition or subpartition name
- Tablespace_name: Tablespace containing the segment
- File_id: Absolute file number of the block in the segment
- Relative_fno: Relative file number of the block in the segment
- Block_id: Block number of the block
- Writetime: Last modification time of the block

Monitoring Statistics: Extent Level

After rows are modified:

- View aggregates at extent level using `DBMS_HEAT_MAP.EXTENT_HEAT_MAP` function:
 - Minimum modification time
 - Maximum modification time

```
SQL> SELECT segment_name, block_id, blocks, max_writetime
  2  FROM table(dbms_heat_map.extent_heat_map
  3                               ('SCOTT','EMPLOYEE'));
```

SEGMENT_	BLOCK_ID	BLOCKS	MAX_WRITETIME
EMPLOYEE	136	8	12-DEC-12 12:58:46
EMPLOYEE	144	8	12-DEC-12 12:58:46
EMPLOYEE	256	128	12-DEC-12 12:58:47
EMPLOYEE	384	128	12-DEC-12 12:58:47

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

`DBMS_HEAT_MAP.EXTENT_HEAT_MAP` Table Function

The `DBMS_HEAT_MAP.EXTENT_HEAT_MAP` function returns the extent-level heatmap for a specific table segment. Aggregates at extent level including minimum modification time and maximum modification time are returned.

The input parameters are the owner of the segment, the `segment_name` and the `partition_name`.

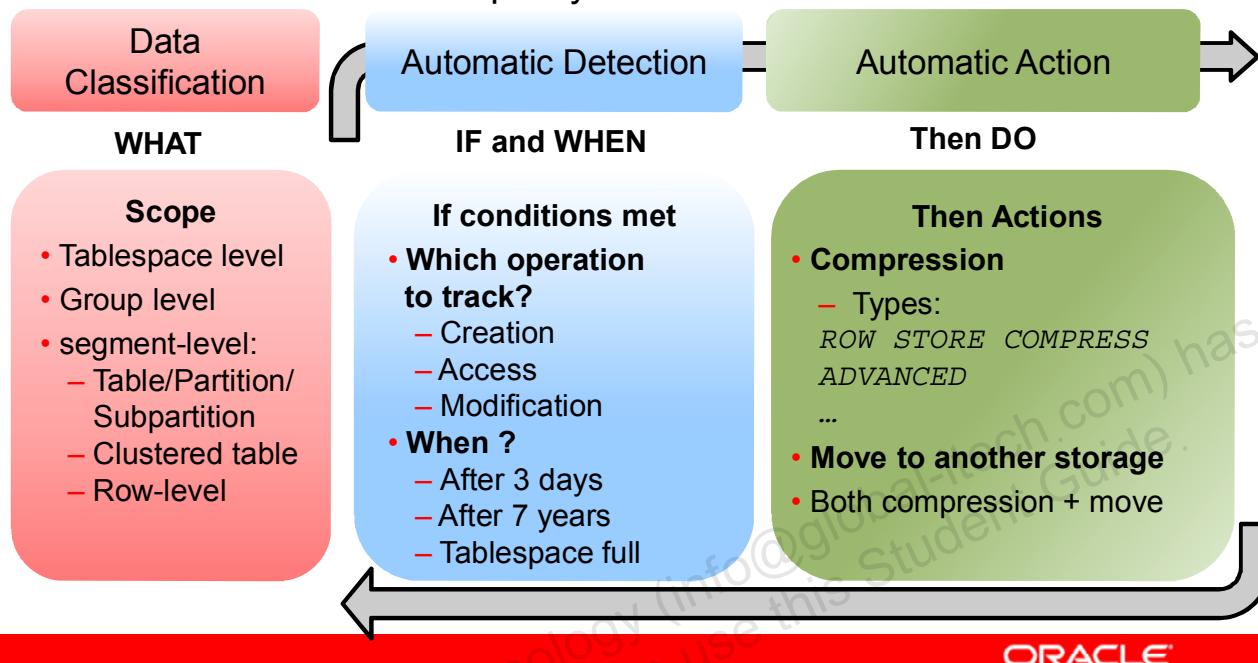
The output parameters are the following:

- Owner: Owner of the segment
- Segment_name: Segment name of the non partitioned table
- Partition_name: Partition or subpartition name
- Tablespace_name: Tablespace containing the segment
- File_id: Absolute file number of the block in the segment
- Relative_fno: Relative file number of the block in the segment
- Block_id: Block number of the block
- Blocks: Number of blocks in the extent
- Bytes: Number of bytes in the extent
- Min_writetime: Minimum of last modification time of the block
- Max_writetime: Maximum of last modification time of the block
- Avg_writetime: Average of last modification time of the block

Defining Automatic Detection Conditions

2. Create ADO policy:

- Define **WHICH** condition fires action.
- Define **WHEN** the policy takes effect.



ORACLE

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

The second step is to create the ADO policy to define the automatic detection:

- Define WHICH condition would fire a compress or move action:
 - Low data access
 - No access
 - No data modification (updates/deletes/inserts on row or DDL ALTER statement on a segment)
 - Object/row creation
 - Tablespace fullness

When a policy is specified on access time, a metric is computed based on the following statistics: Number of accesses by RowID, number of full table scans and number of DML statements in the time period specified by the policy. For example, if the policy mentions `AFTER 3 MONTHS OF LOW ACCESS`, the statistics in the last 3 months are examined to evaluate the low access metric. The metric would weight these statistics based on the source and target compression levels of the policy. For example, if the target compression is `ARCHIVE HIGH` then the metric would give a very high weightage to DML statements and row lookups as opposed to full table scans.

- Define WHEN the policy takes effect: For example, after *n* days or months or years of low or no data access; or no more modification, or after object creation, or when the tablespace containing the object meets the tablespace fullness threshold

Defining Automatic Actions

2. Create ADO policy:

- Define action and level of execution.

	No modification	Creation	Low Access	No Access	Tablespace fullness	Custom policy
Compression (Row)	✓					
Compression (Segment)	✓		✓	✓		✓
Compression (Group)	✓		✓	✓		
Compression (Tablespace)	✓	✓	✓	✓		
Storage Tiering (Segment)					✓	✓
Storage Tiering (Tablespace)					✓	

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

Still in the second step, define the action and the level of automatic execution:

- The action executed (see possible actions in the first column of the table in the slide):
 - Compression and which type of compression
 - Data movement to other storage tier
 - Both when defining two policies on the same segment
- The possible levels of execution:
 - ROW: Row-level ADO policies can only be created based on modification time.
 - SEGMENT: Segment-level ADO policies can apply to tables or partitions.
 - GROUP : Group-level ADO policies indicate that the table's SecureFiles LOBs are compressed as well. Global indexes are maintained. An ADO policy can be specified for a table with the GROUP keyword that is part of the POLICY clause for compression. If the table becomes eligible for an ADO action at any time, then the same ADO action would be performed on all the SecureFiles LOBs of the table as well. For example, if the ADO action is compression, the dependent objects like SecureFiles LOBs would be compressed at compression levels corresponding to a default mapping between heap segment compression levels and those of SecureFiles LOBs. Similar semantics hold for the GROUP keyword for ADO policies on table partitions.
 - TABLESPACE: A DEFAULT ADO policy defined on a tablespace applies to all segments that will be created in the tablespace.

Compression Levels and Types

Compression Level

- Tablespace
- Group
- Segment
- Row: only `ROW STORE COMPRESS [BASIC|ADVANCED]`

Compression Type

- `ROW STORE COMPRESS [BASIC]`
- `ROW STORE COMPRESS ADVANCED`
- `COLUMN STORE COMPRESS FOR QUERY LOW / HIGH`
- `COLUMN STORE COMPRESS FOR ARCHIVE LOW / HIGH`



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

The first possible action is compression. A few reminders about compression: Compression can occur while data is being inserted, updated, or bulk-loaded into a table.

- `ROW STORE COMPRESS BASIC` or `ADVANCED` is used for rows inserted without using direct-path insert and updated rows, using the Advanced Compression option (ACO). `ROW STORE COMPRESS ADVANCED` is the new syntax used with the “Advanced Row Compression” feature, new name for the old OLTP Table Compression feature part of ACO. `ROW STORE COMPRESS ADVANCED` on the heap table maps to `LOW` for SecureFiles LOB segments when the `GROUP` keyword is used.
- `COLUMN STORE COMPRESS FOR QUERY LOW` or `HIGH` provides a higher level of compression than `ROW STORE` compression. It works well when load performance is critical, frequent queries are run against the table, and no normal DML is expected. Column Store is a feature commonly referred to as Columnar Compression or just Columnar or HCC. `COLUMN STORE COMPRESS FOR QUERY LOW/QUERY HIGH` on a heap table maps to `MEDIUM` for SecureFiles LOB segments.
- `COLUMN STORE COMPRESS FOR ARCHIVE LOW/ARCHIVE HIGH` compression provides the highest level of compression and works well for infrequently accessed data, mostly for read-only data. It enables HCC. `COLUMN STORE COMPRESS FOR ARCHIVE LOW/ARCHIVE HIGH` on a heap table maps to `MEDIUM` for SecureFiles LOB segments.

Creating Compression Policies Tablespace and Group

- TABLESPACE level compression policy:

```
SQL> ALTER TABLESPACE tbs1 DEFAULT ILM ADD POLICY
  2          ROW STORE COMPRESS ADVANCED
  3          SEGMENT AFTER 30 DAYS OF LOW ACCESS;
```

- GROUP level compression policy:

```
SQL> ALTER TABLE tab1 ILM ADD POLICY
  2          ROW STORE COMPRESS ADVANCED
  3          GROUP AFTER 90 DAYS OF NO MODIFICATION;
```

```
SQL> ALTER TABLE tab2 MODIFY PARTITION p1 ILM ADD POLICY
  2          COLUMN STORE COMPRESS FOR ARCHIVE HIGH
  3          GROUP AFTER 6 MONTHS OF NO ACCESS;
```



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

All the examples shown in the slide assume that Heat Map is enabled at the instance or session scope.

The first example sets a default tablespace ADO policy so that segments stored in the tablespace inherit the automatic compression with level ROW STORE COMPRESS ADVANCED after 30 days of low access.

In the second and third examples, since the GROUP keyword is specified, dependent SecureFiles LOBs are compressed as well. Global indexes are maintained.

The second example will automatically compress the table and SecureFiles LOBS after 90 days of no modification, with the ROW STORE COMPRESS ADVANCED compression level.

The third example will automatically compress the partition P1 and SecureFiles LOBS after six months of no access, with the level COLUMN STORE COMPRESS FOR ARCHIVE HIGH compression level.

What does LOW ACCESS mean? When a policy condition relies on access time, a metric is computed based on the following statistics in the time period specified by the policy:

- Number of accesses by RowID
- Number of full table scans
- Number of DMLs

For example, if the policy mentions AFTER 3 MONTHS OF LOW ACCESS, statistics in the last 3 months determine the low access metric. The metric would weight these statistics based on the source and target compression levels of the policy. If the target compression is ARCHIVE HIGH, the metric would give a very high weightage to DMLs and row lookups as opposed to full table scans.

Creating Compression Policies

Segment and Row

- **SEGMENT-level compression policy:**

```
SQL> ALTER TABLE tab4 ILM ADD POLICY
  2       COLUMN STORE COMPRESS FOR QUERY HIGH
  3       SEGMENT AFTER 90 DAYS OF NO MODIFICATION;
```

```
SQL> ALTER TABLE tab5 ILM ADD POLICY
  2       COLUMN STORE COMPRESS FOR ARCHIVE HIGH
  3       SEGMENT AFTER 6 MONTHS OF LOW ACCESS;
```

```
SQL> ALTER TABLE tab6 ILM ADD POLICY
  2       ROW STORE COMPRESS ADVANCED
  3       SEGMENT AFTER 6 MONTHS OF NO ACCESS;
```

- **ROW-level compression policy:**

```
SQL> ALTER TABLE tab1 ILM ADD POLICY
  2       ROW STORE COMPRESS ADVANCED
  3       ROW AFTER 30 DAYS OF NO MODIFICATION;
```

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

The first example will automatically compress the table as soon as there is no modification on the segment within 90 days, with the COLUMN STORE COMPRESS FOR QUERY HIGH compression level.

The second example will automatically compress the table as soon as the computed metric for LOW ACCESS on the segment is reached after six months, with COLUMN STORE COMPRESS FOR ARCHIVE HIGH compression level.

The third example will automatically compress the table after six months of no access, with the ROW STORE COMPRESS ADVANCED compression level.

The fourth example will automatically compress any block in which all the row have not been modified in the last 30 days. The only possible compression level at ROW-level is ROW STORE COMPRESS ADVANCED.

Note:

There is no conflict if you have Advanced Row Compression set on a table, and also have ADO policies defined to compress the segment or blocks on policy conditions.

When rows are inserted into the table, they are compressed with Advanced Row Compression.

If your ADO policy says to compress for one of the HCC choices at the segment-level, then that compression is executed as expected when the partition/table qualifies.

If your ADO policy says to perform a Row Compress at the row (block) level, then that compression does not execute because the rows in each block are already compressed - unless you have done things to cause the rows to be uncompressed, such as an ALTER TABLE ... MOVE PARTITION to uncompress all the data in a partition.

Creating Storage Tiering Policy

Movement Level

- Segment-level

Storage Tiering

- Tablespace

```
SQL> ALTER TABLE tab6 MODIFY PARTITION p1 ILM ADD POLICY
2          TIER TO low_cost_storage;
```

Source tablespace fullness thresholds:

```
SQL> SELECT * FROM dba_ilmparameters;
```

NAME	VALUE
<hr/>	
TBS PERCENT USED	85
TBS PERCENT FREE	25

```
SQL> EXEC
DBMS_ILM_ADMIN.CUSTOMIZE_ILM(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,15)
```

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

The second possible action is data movement to another storage tier, whether it is a lower cost storage tier, or a higher-performance storage tier with other compression capabilities such as HCC. The only possible scope for data movement is SEGMENT.

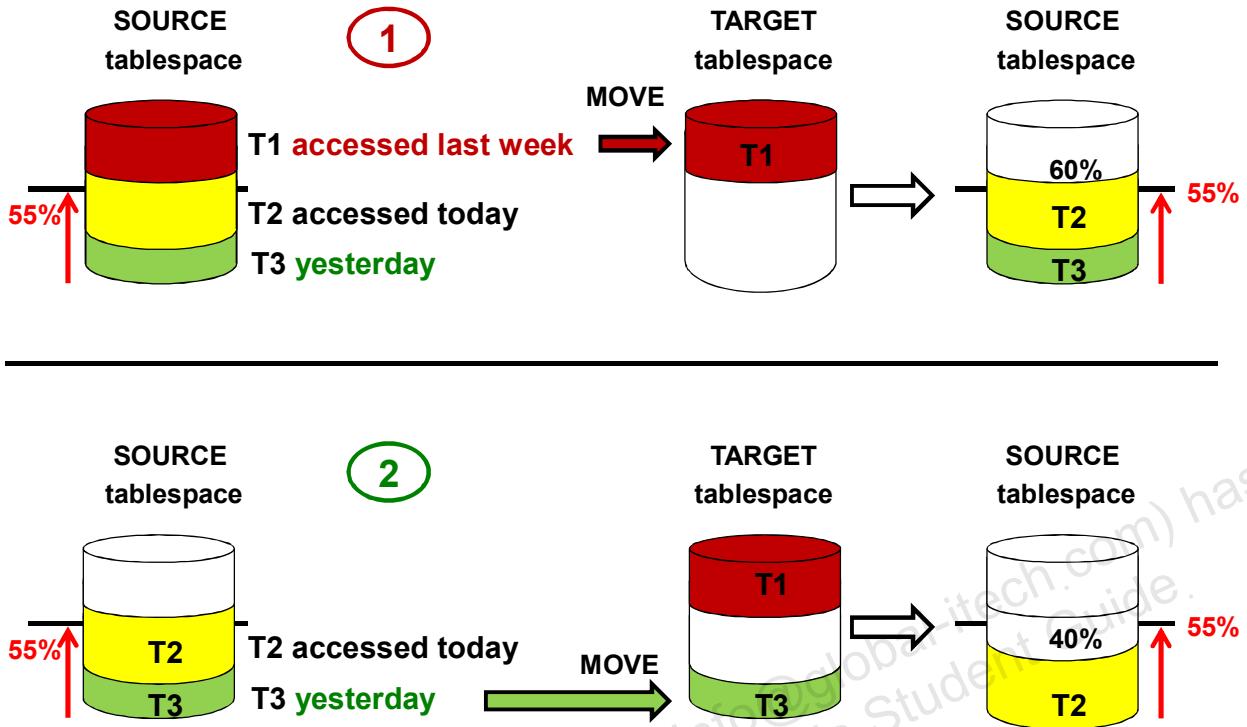
The example shown on the slide automatically moves the partition P1 of the table TAB6 to the LOW_COST_STORAGE tablespace when the source tablespace that contains the partition has reached the fullness threshold.

The tiering fullness threshold of the source tablespace depends on two parameters:

- Objects with tiering policy will be moved if the tablespace they reside in becomes TBS PERCENT USED full (defaulted to 85).
- Objects will be moved to the target tablespace until the source tablespace becomes TBS PERCENT FREE (defaulted to 25).
- Both values can be controlled via the DBMS_ILM_ADMIN.CUSTOMIZE_ILM procedure and displayed in the DBA_ILMPARAMETERS view.

A segment-level policy is executed only once in the lifetime of the policy. After the first execution, the policy is disabled. This applies only to tiering policies.

Storage Tiering: Priority



ORACLE

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

TBS PERCENT USED and TBS PERCENT FREE defaulted to 85 and 25, respectively. This means that, whenever the source tablespace's usage percent goes beyond 85%, any tiering policy specified on its objects will be executed, and all objects will be moved to the target tablespace (because the source tablespace should be 25% free).

Now if the first parameter is set to 55%, such as in our example, which table would move first if several tables of the same tablespace are under the tiering policy? The least recently accessed table will move first.

Therefore, in the slide, the T1 table, accessed last week, moves the first to the target tablespace defined in the tiering policy. The space used in the source tablespace is still above the TBS PERCENT USED value of 55%, and therefore another table needs to move to the target tablespace. T3 accessed yesterday is elected to move to the target tablespace. Now that T1 and T3 freed enough space in the source tablespace to get a TBS PERCENT USED value below 55% and a TBS PERCENT FREE value above 45%, table T2 can remain in the source tablespace.

Storage Tiering: **READ ONLY**

READ ONLY tier storage:

```
SQL> ALTER TABLE tab7 ILM ADD POLICY  
2          TIER TO tablespace_tbs  
3          READ ONLY;
```

```
SQL> ALTER TABLE sales MODIFY PARTITION HY_2010 ILM ADD POLICY  
2          TIER TO tablespace_tbs  
3          READ ONLY;
```



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

When a segment is moved to another tablespace, another option is to set the target tablespace to **READ ONLY** after the object is moved. This is very beneficial during backup, because Oracle Recovery Manager (RMAN) has the ability to skip such tablespaces.

Policy Relying on Function

Customized action:

1. Create a function returning TRUE/FALSE:

```
SQL> CREATE FUNCTION CUSTOM_ILM_RULES
  2      (objn IN NUMBER)
  3      RETURN BOOLEAN ...
n  /
```

2. Use the function in the policy declaration:

```
SQL> ALTER TABLE EMP    ILM ADD POLICY
  2                      ROW STORE COMPRESS ADVANCED
  3                      SEGMENT ON CUSTOM_ILM_RULES;
```

```
SQL> ALTER TABLE sales MODIFY PARTITION before_jan_2005
  2                      ILM ADD POLICY
  3                      TIER TO history ON CUSTOM_ILM_RULES;
```



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

If users need more flexibility to tailor an ADO policy, they can do so with a custom ADO policy. Custom ADO policies utilize a user-provided function to evaluate the policy.

The PL/SQL function encapsulates any business logic required to make a decision about ADO on an object, returning a BOOLEAN value or a number.

The ADO policy will be either enforced or ignored according to the BOOLEAN returned. If the value returned is TRUE, the policy will be executed. If the value returned is FALSE, no action will be taken until the next time the policy is evaluated for execution.

The syntax for custom and non-custom policies cannot be mixed.

Customized policies can be used only with segment-level policies.

Take the second example: When the BEFORE_JAN_2005 table partition contains fewer rows than a defined threshold a certain elapsed time after its creation, you decide to move the segment to an archive storage tier. The rules are described in the function.

Multiple **SEGMENT** Policies on a Segment

New data loaded: **cycle starts**

After 1 day: **Advanced Row compression**

After 1 month: **Query High Compression**

After 1 year: **Move to low cost tablespace**

After 2 years: **cycle ends: purge**

```
SQL> ALTER TABLE t1 ILM ADD POLICY ROW STORE COMPRESS ADVANCED
  2   SEGMENT AFTER 1 DAY OF NO MODIFICATION;
```

```
SQL> ALTER TABLE t1 ILM ADD POLICY
  2   COLUMN STORE COMPRESS FOR QUERY HIGH
  3   SEGMENT AFTER 1 MONTH OF NO MODIFICATION;
```

```
SQL> ALTER TABLE t1 ILM ADD POLICY
  2           TIER TO history ON t1_check_after_one_year;
```

```
SQL> ALTER TABLE t1 ILM ADD POLICY COLUMN STORE COMPRESS FOR
  2           ARCHIVE HIGH SEGMENT AFTER 1 YEAR OF NO ACCESS;
```

```
SQL> DROP TABLE t1 PURGE;
```

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

Several policies can be applied on the same segment as long as they do not have conflicting conditions. Otherwise, the presence of another policy on the same object can disallow certain policies from being applied on the same object.

Conflicts appear if all policies on the same (object, action) are not on the same statistic. For example, all compression policies have to be applied on the same statistic, either access time, modification time or creation time.

It would not make sense to have an Advanced Row compression policy to compress after 3 days of no modification, and also have a QUERY HIGH compression policy to compress after 3 days of no modification. What would make sense is to have an Advanced Row compression policy to compress after 1 day of creation, then have a QUERY HIGH compression policy to compress after 30 days of no modification, then have a ARCHIVE HIGH compression policy to compress after 180 days of no access.

Greater compression levels have to be specified at greater elapsed times.

In this example, the life cycle starts with data load and ends with the table purge operation.

- After one day, ADO evaluates if there were any modification at the table level. If there were none, it proceeds with Advanced Row compression, the lowest level of compression.
- After one month, ADO evaluates if there were any other modification still at the table level. If there were none, it proceeds with QUERY HIGH compression, a higher level of compression than Advanced Row compression and with a greater elapsed time.
- After one year, ADO evaluates if there were any modification. Setting the tablespace fullness threshold to 0, it necessarily moves the segment to the HISTORY tablespace.

After a segment has been moved to a particular compression level, all segment-level policies at a lower compression level are disabled. That is why a segment can move to greater compression levels only via the ILM framework. Row-level policies are not affected by this rule. Also, a compression policy would be executed only if it moves the object to a higher compression level than the dictionary compression level of the object at the time of policy evaluation.

Only One Single **ROW** Policy on a Segment

- Compression level (Row Store Compress) and scope (row): only 1 per object

```
SQL> ALTER TABLE t2 ILM ADD POLICY ROW STORE COMPRESS ADVANCED
  2
  2           ROW AFTER 2 DAYS OF NO MODIFICATION;
```

```
SQL> ALTER TABLE t2 ILM ADD POLICY
  2           ROW STORE COMPRESS ADVANCED INPLACE
  3           ROW AFTER 6 MONTHS OF NO MODIFICATION;
```

- Segment-level policies do not override row-level policies

```
SQL> ALTER TABLE t2 ILM ADD POLICY COLUMN STORE COMPRESS FOR
  2           QUERY HIGH SEGMENT AFTER 1 YEAR OF NO MODIFICATION;
```

```
SQL> ALTER TABLE t2 ILM ADD POLICY
  2           COLUMN STORE COMPRESS FOR ARCHIVE HIGH
  3           SEGMENT AFTER 6 YEARS OF NO ACCESS;
```

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

- Only one (action, level, scope) is allowed on an object. This rule makes sense only for compression policies, where the action is COMPRESSION, the level is the compression level (ROW STORE COMPRESS), and the scope is either ROW or SEGMENT.
The second statement fails at policy creation because a COMPRESSION policy is already configured on the same scope (ROW). The same error is issued if the scope is SEGMENT with the same compression level.
- SEGMENT-level policies do not necessarily override the row-level policies. The third statement succeeds because it configures a COMPRESSION policy with a different compression level on the same statistic (NO MODIFICATION). The forth example fails because it configures a COMPRESSION policy with a different compression level but on a different statistic (NO ACCESS). Another segment tiering policy could be set.

Policy Inheritance

- The child-level policy overrides the parent-level policy with the same action.
- The table-level policy:
 - Overrides the tablespace-level policy
 - Is inherited at the partition-level
- Inheritance is additive for different actions in policies.

```
SQL> select POLICY_NAME "POLICY", OBJECT_NAME,
  2        SUBOBJECT_NAME "SUBOBJECT", OBJECT_TYPE, INHERITED_FROM
  3   FROM DBA_ILMOBJECTS;
```

POLICY	OBJECT_NAME	SUBOBJECT	OBJECT_TYPE	INHERITED_FROM
P281	T1		TABLE	TABLESPACE
P341	SALES		TABLE	POLICY NOT INHERITED
P341	SALES	SALES_1994	TABLE PARTITION	TABLE
P341	SALES	SALES_1996	TABLE PARTITION	TABLE
P350	T3		TABLE	POLICY NOT INHERITED
P360	SALES	SALES_1995	TABLE PARTITION	POLICY NOT INHERITED
P610	T4		TABLE	POLICY NOT INHERITED
P281	T5		TABLE	TABLESPACE

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

When you implement ADO policies on a segment, the following inheritance rules apply:

- For example, if there is a compression policy at the tablespace-level and a compression policy directly specified on an object, the tablespace-level policy would be overridden.
- The nearest ancestor overrides the others if the action is the same. For example, if POLICY1 is specified at the tablespace-level, and POLICY2 at the table level, POLICY2 (which is the nearest ancestor) will be inherited at the partition-level, if the action specified by both policies is the same.
- Inheritance is additive for different actions in policies.

The DBA_ILMOBJECTS view shows all policies inherited by an object. They also have a field to indicate the level from which the policies were inherited.

In the example, tables T1 and T5 inherited the P281 policy from the tablespace they reside in, the SALES_1995 table partition did not inherit the P360 policy from the SALES table. The policy was directly applied to the table partition, whereas the SALES_1994 and SALES_1996 table partitions inherited the P341 policy from the SALES table.

Displaying Policies

DBA_ILMPOLICIES/DBA_ILMDATAMOVEMENTPOLICIES

```
SQL> SELECT * FROM DBA_ILMPOLICIES;
```

POLICY_NAME	POLICY_TYPE	TABLESPACE	ENABLED
P281	DATA MOVEMENT		YES
P381	DATA MOVEMENT		YES
P400	DATA MOVEMENT	ITB	YES

```
SQL> SELECT policy_name, action_type,
  2      compression_level, tier_tablespace TBS
  3  FROM    DBA_ILMDATAMOVEMENTPOLICIES;
```

POLICY_NAME	ACTION_TYPE	COMPRESSION_LEVEL	TBS
P281	COMPRESSION	QUERY HIGH	
P381	COMPRESSION	ADVANCED	
P400	STORAGE		ITB

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

After ADO policy implementation, use views to:

- Verify the configured policies.
- Monitor ADO to verify that data movement and compression took place automatically when desired.

DBA_ILMPOLICIES provides the list of all policies and their respective ENABLED status.

An interesting view is DBA_ILMDATAMOVEMENTPOLICIES.

In the example, two compression policies are created with different compression levels, and a third TIER_TO policy to move a segment to the ITB tablespace under certain conditions. The conditions are stored in the CONDITION_TYPE and CONDITION_DAYS columns.

Displaying Policies

DBA_ILMDATAMOVEMENTPOLICIES

POL	ACTION_TYPE	SCOPE	CONDITION_TYPE	CONDITION_DAYS
P341	COMPRESSION	GROUP	LAST MODIFICATION TIME	90
P342	COMPRESSION	ROW	LAST MODIFICATION TIME	30
P361	COMPRESSION	GROUP	LAST ACCESS TIME	180
P381	COMPRESSION	SEGMENT	LAST MODIFICATION TIME	90
P382	COMPRESSION	SEGMENT	LOW ACCESS	180
P401	COMPRESSION	SEGMENT	LAST ACCESS TIME	180
P402	STORAGE	SEGMENT		0
P321	COMPRESSION	SEGMENT	LOW ACCESS	30
P461	COMPRESSION	SEGMENT	USER DEFINED	0
P462	STORAGE	SEGMENT	USER DEFINED	0
P482	COMPRESSION	SEGMENT	LAST MODIFICATION TIME	30
P681	COMPRESSION	GROUP	LAST ACCESS TIME	180



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

In the example, a row-level compression policy P342 is created to automatically compress the blocks of a table after 30 days of no modification:

```
SQL> alter table employee ILM ADD POLICY
  2  ROW STORE COMPRESS ADVANCED
  3  ROW after 30 days of no modification;
```

CONDITION_TYPE represents the statistic type.

Preparing Evaluation and Execution

Segment-level policy	Row-level policy
In maintenance window	Regularly by MMON every 15 minutes

- Change the interval for ADO policies evaluation.

```
SQL> EXEC DBMS_ILM_ADMIN.CUSTOMIZE_ILM(
      DBMS_ILM_ADMIN.EXECUTION_INTERVAL, 1)
```

- Two ways for ADO execution with **DBMS_ILM** package.
 - Immediate ADO task execution on a set of objects.

```
SQL> VAR v_taskid NUMBER
SQL> EXEC DBMS_ILM.EXECUTE_ILM (owner =>'SCOTT', object_name
=>'EMPLOYEE', task_id =>:v_taskid)
```

- Customize the set of objects before ADO task execution.
- Stop execution.

```
SQL> EXEC DBMS_ILM.STOP_ILM(245321)
```

ORACLE

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

It is sometimes necessary to move data as quickly as possible from one tier to another or compress data as soon as possible, and it is not possible to wait until the next maintenance window.

The **DBMS_ILM** package supports immediate evaluation and immediate/background scheduling of execution of ADO related tasks. The **DBMS_ILM** package supports the following two ways for scheduling ADO related tasks execution.

- A database user schedules immediate ADO execution on a set of objects. Although you can create a schedule to trigger the ADO actions evaluation more often than it does with the maintenance window, you can start the evaluation now. The **EXECUTE_ILM** procedure provides that ability, regardless of any previously scheduled ADO execution.
- A database user can view first view the results of evaluation of ILM policies on a set of objects before ADO related tasks execution and customize the list of objects candidate for ADO execution.

It is also possible to stop a particular execution.

Customizing Evaluation and Execution

- Use DBMS_ILM package and procedures to:
 - Customize the set of objects to evaluate
 - Schedule execution of ADO tasks



```
SQL> EXEC DBMS_ILM.PREVIEW_ILM (TASK_ID => :v_taskid, -
  2                               ILM_SCOPE => DBMS_ILM.SCPE_SCHEMA)
```

```
SQL> EXEC DBMS_ILM.EXECUTE_ILM_TASK (TASK_ID => 26382, -
  2     EXECUTION_MODE => DBMS_ILM.SCHEDULE_IMMEDIATE, -
  3     EXECUTION_SCHEDULE => DBMS_ILM.ILM_EXECUTION_OFFLINE)
```

```
SQL> SELECT task_id, state FROM dba_ilmtasks;
```

TASK_ID	STATE
26482	ACTIVE

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

Use the DBMS_ILM.PREVIEW_ILM procedure to view the results of evaluation of ADO policies on a set of objects. The scope of evaluation can be one of the two:

- A whole schema: attribute ILM_SCOPE => DBMS_ILM.SCPE_SCHEMA. The evaluation selects all ADO policies in the current schema. (This is the default)
- The whole database: attribute ILM_SCOPE => DBMS_ILM.SCPE_DATABASE. The evaluation selects all ADO policies in the database.

The task created is in INACTIVE state. The user can still add or delete objects to this set and reviews the results of ADO policy evaluation again. The user repeats this step until he/she finalizes the set of objects for ADO execution.

Then use the EXECUTE_ILM_TASK procedure to perform immediate scheduling of ADO execution on this set of objects. The state of the task moves to ACTIVE state when it starts executing. Once the execution of the task is finished, the state becomes COMPLETE.

The execution of the task with EXECUTE_ILM_TASK procedure launches a job executed according to the attributes set in the procedure:

- EXECUTION_MODE: ILM_EXECUTION_ONLINE value executes the task online and ILM_EXECUTION_OFFLINE value executes the task offline.
- EXECUTION_SCHEDULE: Identifies when the ILM task should be executed. The only possible value in Oracle Database 12.1 is DBMS_ILM.SCHEDULE_IMMEDIATE.

Monitoring Evaluation and Execution

- DBA_ILMTASKS
- DBA_ILMEVALUATIONDETAILS
- DBA_ILMRESULTS

```
SQL> SELECT * FROM dba_ilmtasks;
```

TASK_ID	TASK_OWNER	STATE	CREATION_TIME	START_TIME	COMPLETION_TIME
18566	SCOTT	INACTIVE	12-DEC-12		
18542	SCOTT	ACTIVE	11-DEC-12		12-DEC-12

```
SQL> SELECT task_id, policy_name, object_name,
  2      selected_for_execution, job_name
  3  FROM dba_ilmevaluationdetails;
```

TASK_ID	POLICY_NAME	OBJECT_NAME	SELECTED_FOR_EXECUTION	JOB_NAME
18762	P281	EMPLOYEE	POLICY DISABLED	
18542	P281	EMPLOYEE	SELECTED FOR EXECUTION	ILMJOB5002
18862	P301	EMPLOYEE	PRECONDITION NOT SATISFIED	

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

View the results of the ADO tasks and jobs executions in the following views:

- DBA_ILMTASKS contain new information about ADO tasks every time a user invokes the procedures EXECUTE_ILM or EXECUTE_ILM_TASK. A task ID is returned to track this particular invocation. A task ID is also generated to track periodic internal ADO executions by the database. This view contains information on all ADO executions. You can see that all SYS evaluation tasks are regularly executed every 15 minutes.
- DBA_ILMEVALUATIONDETAILS contain details on the tasks execution after the ADO evaluation. The SELECTED_FOR_EXECUTION column informs whether the policy has been selected for execution on the object on which the ADO policy exists. A “SELECTED FOR EXECUTION” value triggers an ADO job whereas a “PRECONDITION NOT SATISFIED” value does not. This column can take one of the following values: POLICY DISABLED, POLICY OVERRULED, INHERITED POLICY OVERRULED, JOB ALREADY EXISTS, NO OPERATION SINCE LAST ILM ACTION, TARGET COMPRESSION NOT HIGHER THAN CURRENT, STATISTICS NOT AVAILABLE

The first row of the first example shows that an ADO task was in INACTIVE state, probably due to an ADO preview not yet finalized by the user. The second row shows a task being in ACTIVE state with a creation date and a starting date, but not yet completed with a completion date.

The first row of the second example shows that an ADO policy exists on the EMPLOYEE table but is disabled. The second row shows that the same ADO policy had been evaluated positively and generated a job to execute ADO actions. The third row shows that another ADO policy on the same table has not positively passed the evaluation, the conditions under which ADO actions should be triggered are not satisfied.

- DBA_ILMRESULTS provides information about ADO jobs manual or automatic executions.

```
SQL> SELECT task_id, cast(job_name as varchar2(20))  
  2      job_name, job_state, completion_time  
  3  FROM DBA_ILMRESULTS;
```

TASK_ID	JOB_NAME	JOB_STATE	COMPLETION_TIME
666	ILMJOB59	COMPLETED	SUCCESSFULLY 28/09/2011

ADO DDL

- Enable all policies on an object:

```
SQL> ALTER TABLE tab1 ILM ENABLE_ALL;
```

```
SQL> ALTER TABLE tab2 MODIFY PARTITION p1 ILM ENABLE_ALL;
```

- Disable all policies on an object:

```
SQL> ALTER TABLE tab1 ILM DISABLE_ALL;
```

- Delete all policies on an object:

```
SQL> ALTER TABLE tab1 ILM DELETE_ALL;
```

```
SQL> ALTER TABLE tab1 MODIFY PARTITION p1 ILM DELETE_ALL;
```



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

Enabling all ADO policies on a table or partition is done in the first and second examples.

Disabling all ADO policies on a table is done in the third example. Disabling all ADO policies on a table partition would use the MODIFY PARTITION and DISABLE_ALL clauses.

Deleting all ADO policies on a table or partition is done in the fourth and fifth examples.

Turning ADO Off and On

- Turn off ADO without disabling or deleting all policies:

```
SQL> exec DBMS_ILM_ADMIN.DISABLE_ILM
```

- Turn ADO back on for all policies:

```
SQL> exec DBMS_ILM_ADMIN.ENABLE_ILM
```

- View ADO parameters:

```
SQL> select * from DBA_ILMPARAMETERS where name='ENABLED';
```

NAME	VALUE
ENABLED	1



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

You can turn off background ADO by using the `DBMS_ILM_ADMIN.DISABLE_ILM` procedure, and you can turn it on by using `DBMS_ILM_ADMIN.ENABLE_ILM` procedure.

The `ENABLED` attribute is displayed in the `DBA_ILMPARAMETERS` view. The value set to 1 means that ADO is enabled, 2 means that ADO is disabled. Disabling ADO does NOT delete ADO policies nor disable them.

You need the `SYSDBA` privilege to use this package.

Stop Activity Tracking and Clean Up Heat Map Statistics

- Stop activity tracking in session:

```
SQL> ALTER SESSION SET HEAT_MAP = OFF;
```

- Stop activity tracking at instance level:

```
SQL> ALTER SYSTEM SET HEAT_MAP = OFF;
```

- Clean up heat map statistics:

```
SQL> exec DBMS_ILM_ADMIN.CLEAR_HEAT_MAP_ALL;
```



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

You can stop the activity tracking setting the HEAT_MAP parameter to OFF at the session or instance scope.

Disabling Heat Map for a specific session which has Heat Map enabled at instance level allows the subsequent SQL statements for that session not to be tracked by Heat Map.

To clear all heat map statistics in the SYS.HEAT_MAP_STAT\$ table, use the DBMS_ILM_ADMIN.CLEAR_HEAT_MAP_ALL procedure.

Specific Situations of Activity Tracking

- Create the NEW_EMP table from a table with ADO policies:

```
SQL> CREATE TABLE new_emp AS SELECT * FROM emp;
```

- The NEW_EMP table does not inherit the ADO EMP policies.

- Row-level ADO policy job fails before end.
- Segment-level ADO jobs fails before end.



- COMPRESS attribute: compatible with ADO compression policy



```
SQL> CREATE TABLE tabcomp (c number) ROW STORE COMPRESS;
SQL> ALTER TABLE tabcomp ILM ADD POLICY
      2          ROW STORE COMPRESS ADVANCED
      3          ROW AFTER 6 DAYS OF NO MODIFICATION;
```

ORACLE

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

- You create an EMP table with ADO policies and then a NEW_EMP table selecting the structure and rows from the source EMP table. Note that the NEW_EMP table does not inherit the ADO policies of the source table.
- It is possible for ADO jobs to fail before completing their task.
 - In the case of row-level compression ADO policies, each block is compressed in a transaction. If the transaction is in progress when the job is killed, it will be rolled back. If some blocks were already compressed, those changes will not be affected.
 - The case of segment-level ADO jobs is more complex. For example, a job involving ALTER TABLE ... MOVE can fail after moving the segment, but before rebuilding the indexes on the segment. In such cases, jobs will be spawned to rebuild all indexes that were usable before the ALTER TABLE ... MOVE operation started.
- There is no conflict if you have both Advanced Row compression set on a table, and ADO policies causing data to be compressed based on conditions. When rows are inserted into the table, they are compressed with Advanced Row compression. If the ADO policy says to compress for one of the HCC choices at the segment-level, the compression executes as expected when the partition/table qualifies. If the ADO policy says to compress at the row (block) level, the compression does not execute because the rows in each block are already compressed due to the ROW STORE COMPRESS attribute set on the table – unless you have done operations to cause the rows to be uncompressed, such as an ALTER TABLE ... MOVE PARTITION to uncompress all the data in a partition.

Quiz

Which of the following is a limitation in ADO compression?

- a. You cannot compress at block level.
- b. You cannot compress at segment-level.
- c. You can compress at row-level for QUERY LOW.
- d. None of them.



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

Answer: d

Quiz

You can stop activity tracking at the segment-level in your session.

- a. True
- b. False



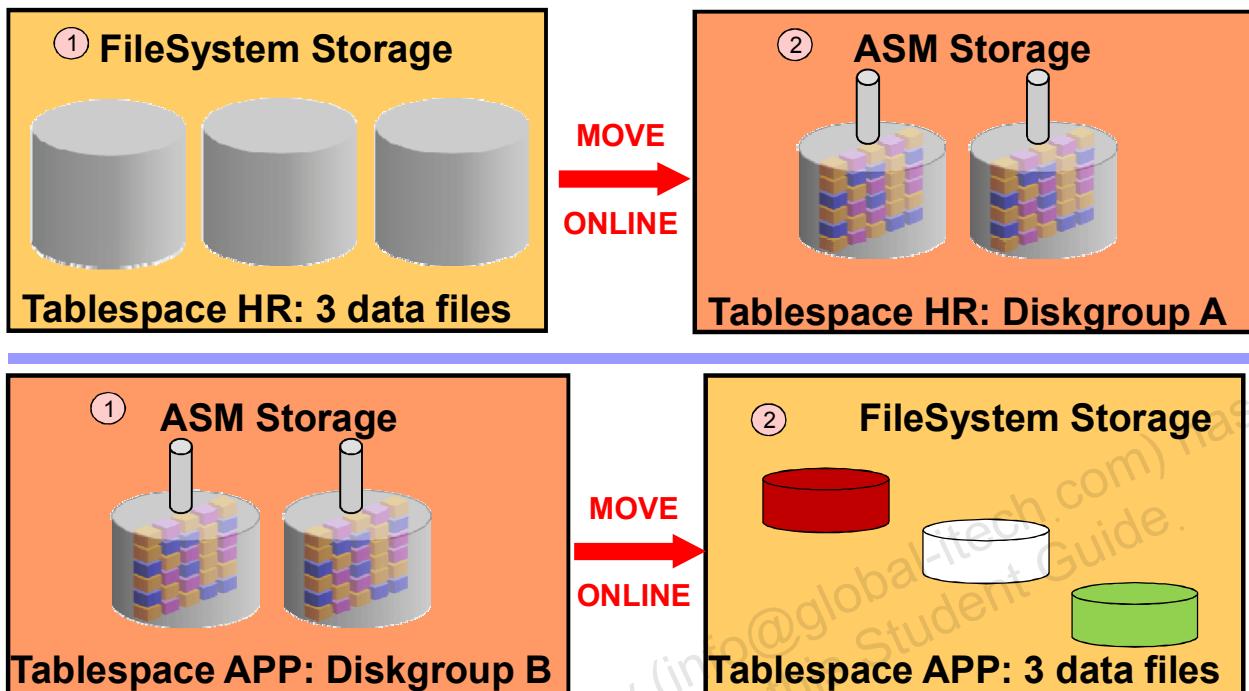
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

The heat map can be stopped at instance or session scope, not at a lower level than these.

Online Move Data File

Move a **data file** to another kind of storage system **ONLINE**.



ORACLE

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

The Oracle Database 12c Online Move data file feature provides the capability to move an online data file from one kind of storage system to another while the database is open and accessing the file.

This implies that queries, DML and DDL operations can be performed while the data file is being moved:

- You can create tables and indexes.
- You can rebuild indexes online.
- You can select tables and partitions data.
- If objects are compressed while the data file is moved, the compression remains the same.

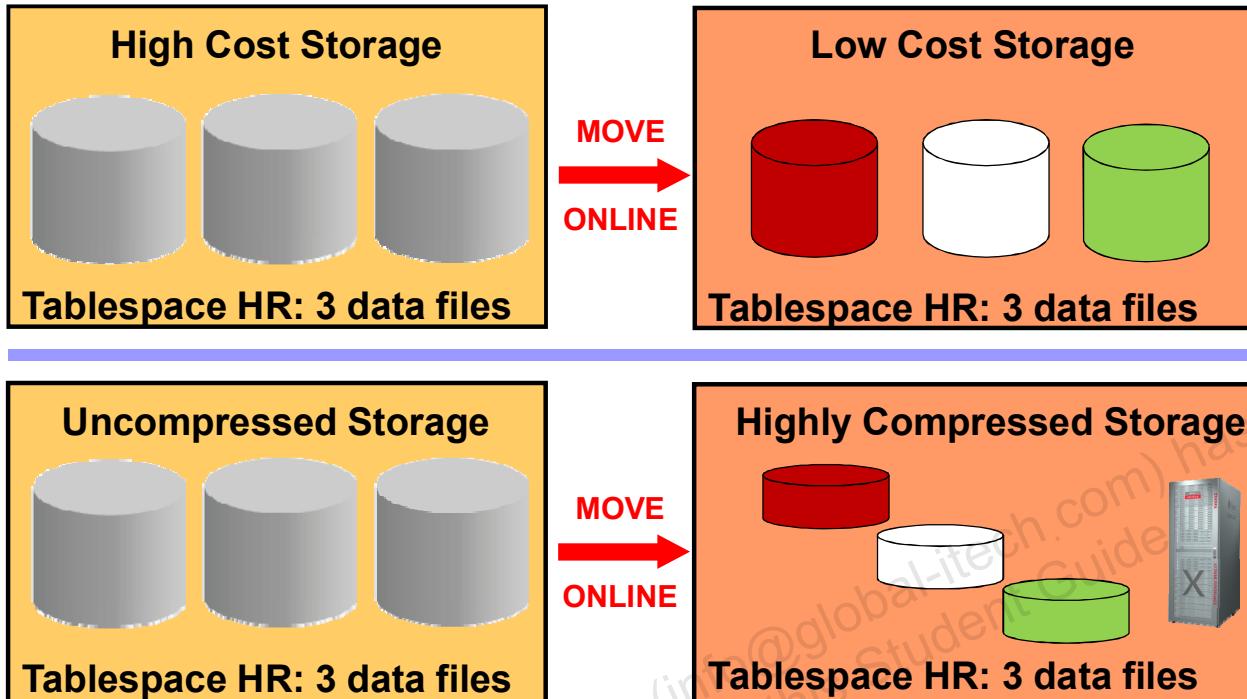
This means that many maintenance operations can be used to move:

- Data files from one kind of storage system to another
- A database into ASM without going down

ADO can benefit from this new enhancement. It allows easy, transparent ADO actions, such as data movement to other storage tier while tablespaces or data files move to another storage tier. Associated segments consequently can move to another storage tier.

Compression

Move **data file** online to lower-cost storage:



ORACLE

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

This means that many maintenance operations can be used to:

- Move tablespaces or data files from a high-cost storage to a lower-cost storage and meanwhile let ADO data movement actions execute.
- Move uncompressed or BASIC/OLTP compressed data to another storage system with a higher compression level, such as HCC. It can be COMPRESSION FOR QUERY for data warehouse performance queries or FOR ARCHIVE for archiving dormant data. The ADO compression action can benefit from this new enhancement.

REUSE and KEEP

No need for database SHUTDOWN / data file OFFLINE

11g

```
SQL> alter database rename file '/u01/t1.f' to '/u02/t1.f';
...
ORA-01511: error in renaming log/data files
ORA-01121: cannot rename database file 5 - file is in use
or recovery
```

12.1

```
SQL> ALTER DATABASE MOVE datafile '/disk1/myexample01.dbf'
2
          TO '/disk2/myexample01.dbf' REUSE;
```

```
SQL> ALTER DATABASE MOVE datafile '/disk1/myexample01.dbf'
2
          TO '+DiskGroup2' KEEP;
```

```
SQL> ALTER DATABASE MOVE datafile 5 TO '+DiskGroup3';
```

ORACLE

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

The advantage is that while you move a data file to another location, disk, or storage system, you do not have to shut down the database or move the data file offline.

While the different operations displayed in the slide are executing, ADO actions, such as compression of segments or data movement, can take place.

The Oracle Database 11g example shows that the file has to be moved offline first.

The Oracle Database 12c example shows how to move a data file from disk1 to disk2.

The `TO` clause can be omitted only when an Oracle-managed file is used. In this case, the `DB_CREATE_FILE_DEST` parameter should be set to indicate the new location.

If the `REUSE` clause is specified, the new file is created even if it already exists. The `REUSE` clause can be used if a previous move command failed and the user reissues the same move command. If the `KEEP` clause is specified, the old file will be kept after the move operation.

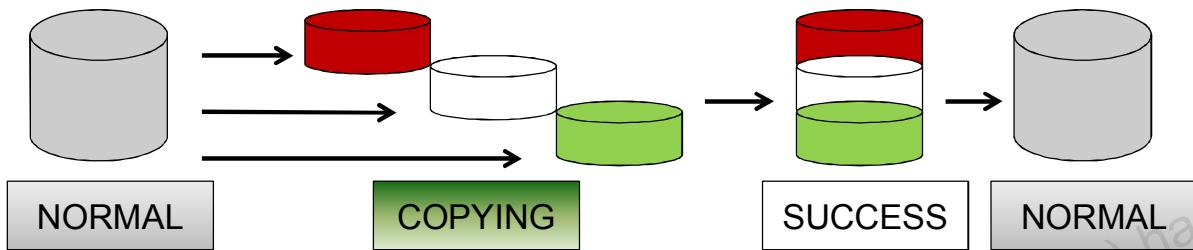
The `KEEP` clause is not allowed if the source file is an Oracle-managed file.

By default, neither `REUSE` nor `KEEP` are used.

States

Progress in V\$SESSION_LONGOPS:

- One row for each file moved
- Number of blocks moved so far



ORACLE

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

The state transition of a successful online move operation is usually NORMAL to COPYING to SUCCESS and finally to NORMAL.

Moving a data file might be a long operation.

Use the V\$SESSION_LONGOPS view to display ongoing online move operations. Each ongoing operation has one row.

Compatibilities

NOT compatible with:	<ul style="list-style-type: none"> • Data file OFFLINE • Concurrent FLASHBACK DATABASE operation • Concurrent media recovery • data file RESIZE (shrink) operation
Compatible with:	<ul style="list-style-type: none"> • Block media recovery • Tablespace made READ ONLY or READ WRITE • data file RESIZE (extension) • Online backup



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

An Online Move data file operation is not compatible when:

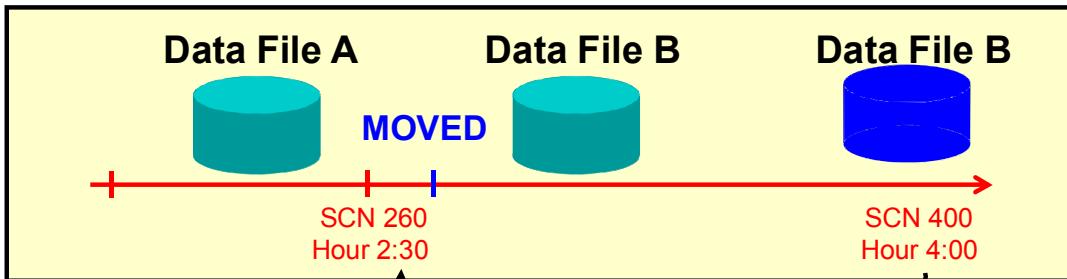
- The data file is an OFFLINE data file
- A concurrent flashback database operation is executing
- A media recovery is completing
- A file shrink operation or tablespace offline/drop operation involving the same file is performing

But it is compatible with:

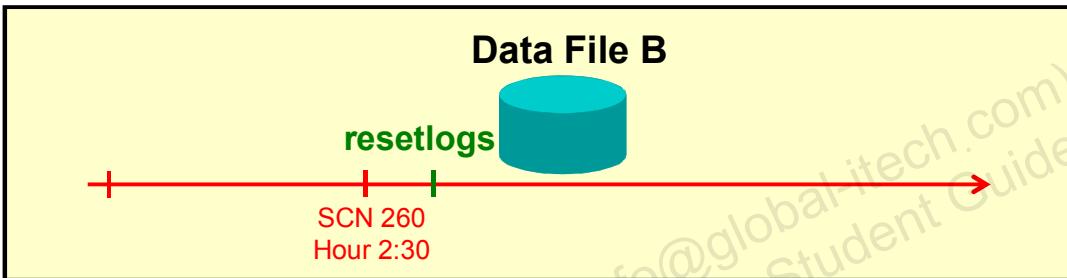
- Block media recovery
- ALTER TABLESPACE READ ONLY or READ WRITE operations
- Data file extension operation
- Tablespace/database online backup mode involving the same file

Flashback Database

Flashback database to SCN 260:



After flashback database:



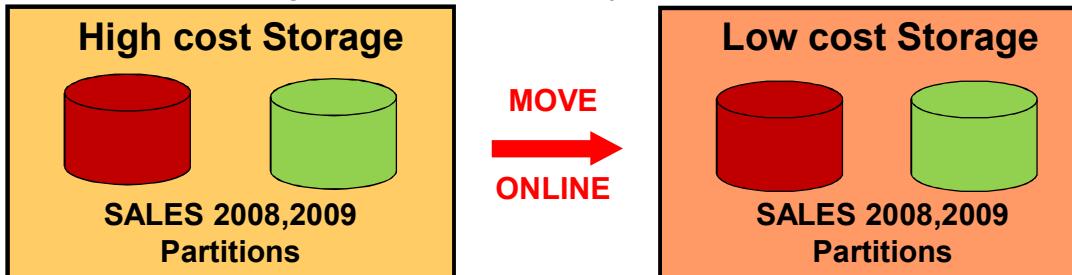
ORACLE

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

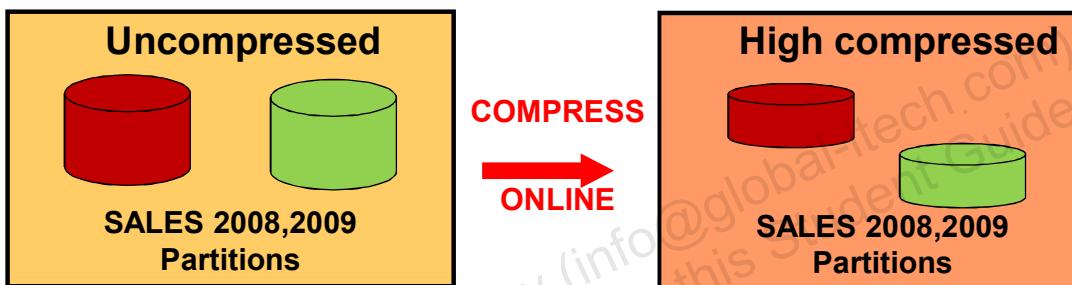
If a flashback database is executed to a time when the file was not yet moved, the flashback database operation will not change the file name to the original name, although it will bring back the old contents of the file.

Online Move Partition

- Move / split / merge partitions and subpartitions **ONLINE** to low cost storage once 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 once they become infrequently accessed, for example according to ADO policies configured
- Compress time based partitions and subpartitions according to ADO policies configured

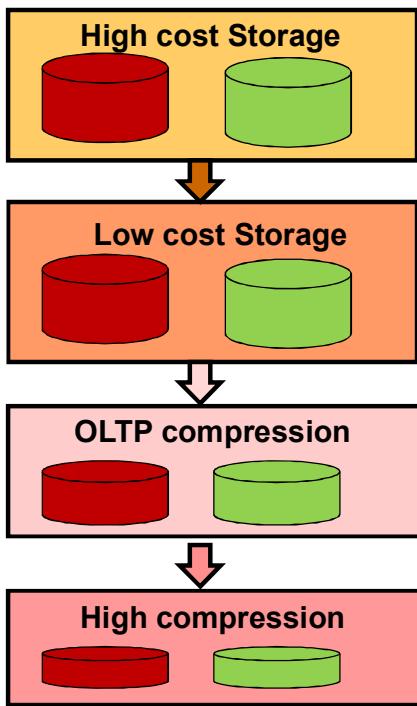
`ALTER TABLE MOVE (SUB) PARTITION` is made online to allow ADO policies to automatically move partitions or change partition compression properties in the background.

ILM can benefit from this new enhancement to move and compress table partitions according to ADO 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 Move Partition: Benefits

SALES old Partitions



- DML allowed
- Move, split or merge partitions ONLINE to low cost storage
- Global and local indexes maintained

```
SQL1> UPDATE ORDERS
      2  SET SAL_ORD_P1=SAL*2
      3  WHERE DATE_P1='2001';
```

```
SQL2> ALTER TABLE ORDERS
      2  MOVE PARTITION ORD_P1
      3  TABLESPACE lowtbs
      4  UPDATE INDEXES ONLINE;
```

- Compress partitions ONLINE

```
SQL> ALTER TABLE ORDERS
      2  MOVE PARTITION ORD_P1
      3  ROW STORE COMPRESS
      4  UPDATE INDEXES ONLINE;
```

ORACLE

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

Oracle now provides an `ALTER TABLE ... MOVE PARTITION ONLINE` option where DML operations can continue to run uninterrupted on the partition that is being moved. In addition, global and local indexes are maintained during the move partition. 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 Move Partition: Compress

MOVE is also used for compression.

- ROW STORE COMPRESS [BASIC]

```
SQL> ALTER TABLE ORDERS MOVE PARTITION ORD_P1
  2      ROW STORE COMPRESS
  3      UPDATE INDEXES ONLINE;
```

- ROW STORE COMPRESS ADVANCED

```
SQL> ALTER TABLE ORDERS MOVE PARTITION ORD_P1
  2      ROW STORE COMPRESS ADVANCED
  3      UPDATE INDEXES ONLINE;
```

- COLUMN STORE COMPRESS

```
SQL> ALTER TABLE ORDERS MOVE PARTITION ORD_P1
  2      COLUMN STORE COMPRESS FOR QUERY HIGH
  3      UPDATE INDEXES ONLINE;
```

```
SQL> ALTER TABLE ORDERS MOVE PARTITION ORD_P1
  2      COLUMN STORE COMPRESS FOR ARCHIVE HIGH
  3      UPDATE INDEXES ONLINE;
```

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

MOVE is typically the vehicle for compressing existing segments, therefore Row Store or Column Store compression types are supported. These are the new compression types of the Advanced Row Compression.

Row Store or Column Store are attributes of a table and partition, like the organization clause. ROW STORE and COMPRESS are the new attributes to compress a table or a partition (ROW STORE versus COLUMN STORE or HCC).

ADVANCED and BASIC are types of compression, used uniformly for tables, LOBs and partitions to convey that it is part of ACO.

The first example shows how to compress with BASIC level an uncompressed partition ORD_P1 online. Global and local indexes are maintained during the partition compression.

The second example shows how to compress with OLTP level an partition ORD_P1 online. Global and local indexes are maintained during the partition compression.

The third and fourth examples show how to compress with HCC level a partition ORD_P1 online. Global and local indexes are maintained during the partition compression.

Quiz

Which of the following is true regarding Online Move Datafile?

- a. Moving a data file online prevents DML operations on segments stored in the data file.
- b. Flashing back a database to a point in time before a data file had been renamed online retains the new name.
- c. Moving a data file online can be performed if the data file is OFFLINE mode.
- d. Moving a data file online cannot be performed if the data file is under BEGIN BACKUP.



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

Answer: b

Summary

In this lesson, you should have learned how to:

- Describe the new Oracle Database 12c Automatic Data Optimization features
- Describe activity tracking with Heat Map
- Use views to display Heat Map statistics
- Describe Automatic Data Optimization
- Create ADO policies
- Use views and procedures to monitor Automatic Data Optimization
- Move a data file online
- Move a partition online



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

Practice 7 Overview: Moving Data Files Online and Practicing ADO

These practices cover the following topics:

- Enabling Heat Map
- Creating ILM TIER TO policy
- Checking automatic data optimization using views
- Creating ILM COMPRESS policy
- Checking automatic data compression
- Moving data files online



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

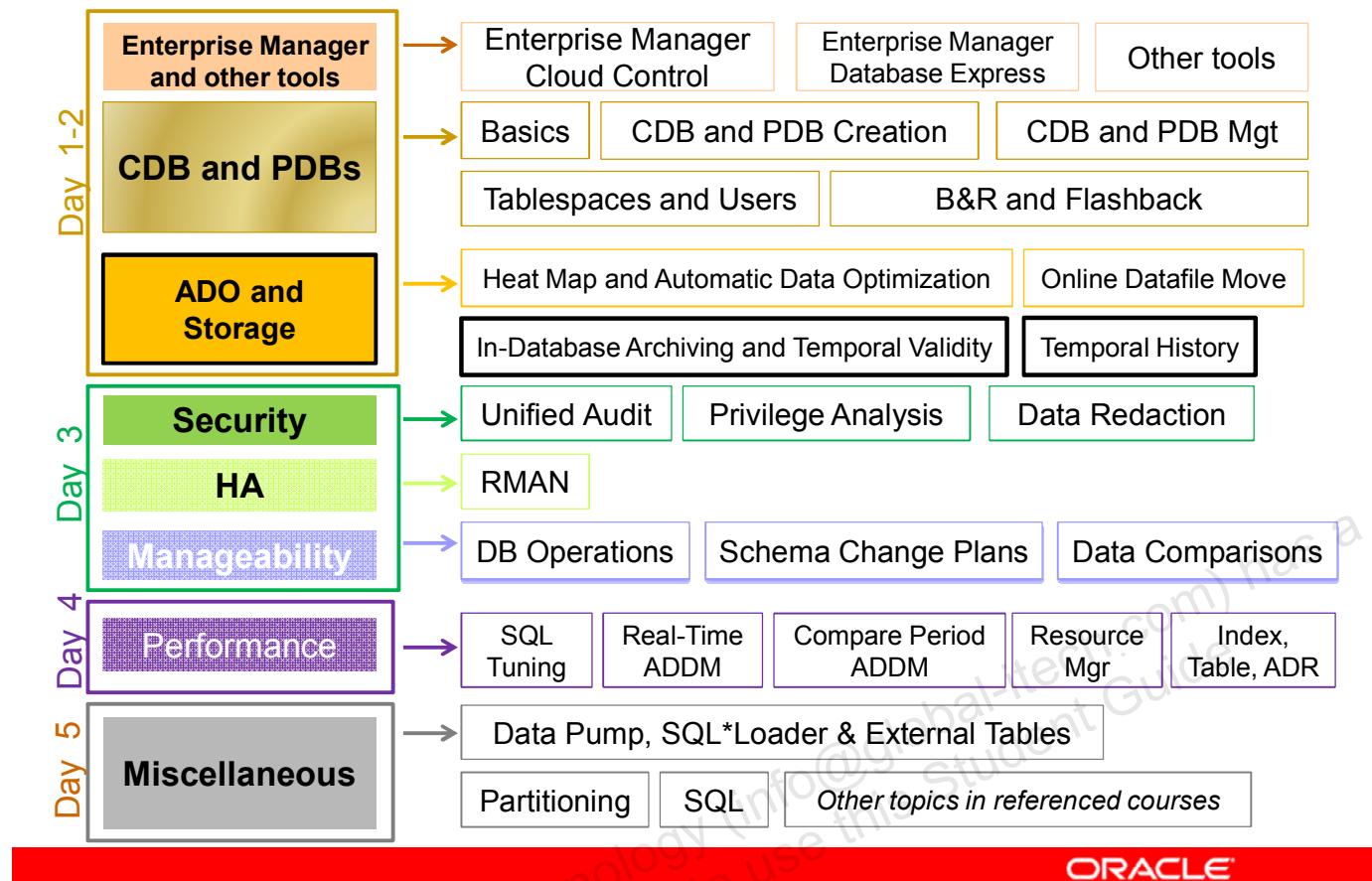
In-Database Archiving and Temporal

8

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.

In-Database Archiving: In-Database Archiving solves problems regarding non-active data. It gives users the ability to keep both active operational data and non-active data in the same database tables. While achieving operational efficiency of archiving, applications can easily be limited to only access data that is in an operationally active state although archived data is kept in the same table.

Temporal Validity: Temporal Validity is a new temporal database capability of Oracle Database 12c. Temporal Validity provides support for valid-time (user-defined time) semantics. This new capability allows you to create a valid time dimension for each row of a table, and is another way of indicating operational relevance of rows.

The Total Recall database option, also known as Flashback Data Archive (FDA) is now called **Temporal History** and is no longer an option. In Oracle Database 12c, it is enhanced with new capabilities.

Objectives

After completing this lesson, you should be able to:

- Distinguish In-Database Archiving from Temporal Validity
- Enable row archival for visibility of active data only
- Describe the advantages of temporal validity versus temporal history
- Set temporal validity at table level
- Use a new SQL temporal data type
- Use temporal validity to reflect business validity
- Understand session-level or flashback point-in-time query
- Describe Temporal History (FDA) enhancements



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

Note: For a complete understanding of Oracle Information Lifecycle Management new features and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database VLDB and Partitioning Guide 12c Release 1 (12.1)*
- *Oracle Database PL/SQL Packages and Types Reference 12c Release 1 (12.1) – “DBMS_FLASHBACK_ARCHIVE” chapter*

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

- *Oracle By Example (OBE):*
 - *Using In-Database Row Archiving*
 - *Implementing Temporal Validity*

Archiving Challenges

Challenges in archiving include:

- Keeping archived data instantly available in the database
- Writing programs to reload data from tape into database
- Backing up huge amounts of data, including old data
- Removing old data



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

There usually comes a point during the life cycle of the data when data is no longer being regularly accessed and is considered eligible for archiving. Traditionally, the data would have been removed from the database and stored on tape, because it can store vast quantities of information for a very low cost. This solution has major disadvantages such as:

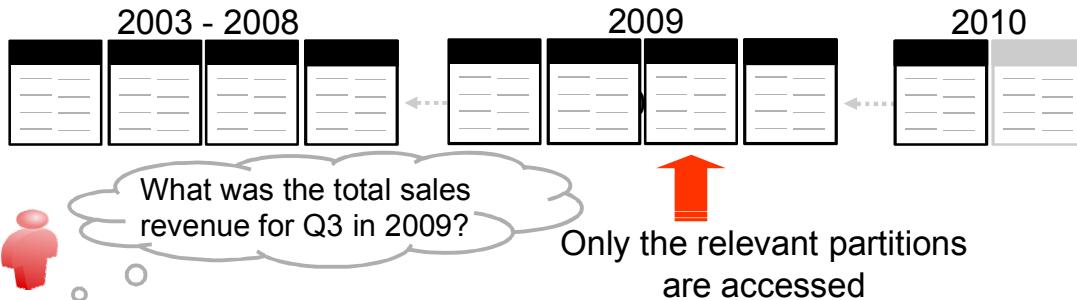
- The data is not instantly available.
- Developers have to write programs to reload the data into the database from the tape archive. This could prove to be expensive and time-consuming.

The historical data is commonly not kept in the database because backing up huge amounts of old data has a major impact on backups.

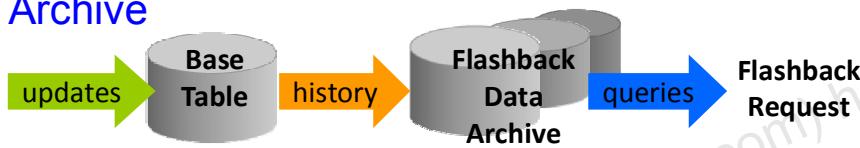
Today in Oracle Database 11g, it is no longer necessary to archive that data to tape, instead it can remain in the database. This is called In-Database Archiving, a mechanism that stores operational and archival data in the same database as opposed to out-of-database archiving of old/historical data.

Archiving Solutions

Partitioning



Flashback Data Archive



ILM Online Archive at SEGMENT level

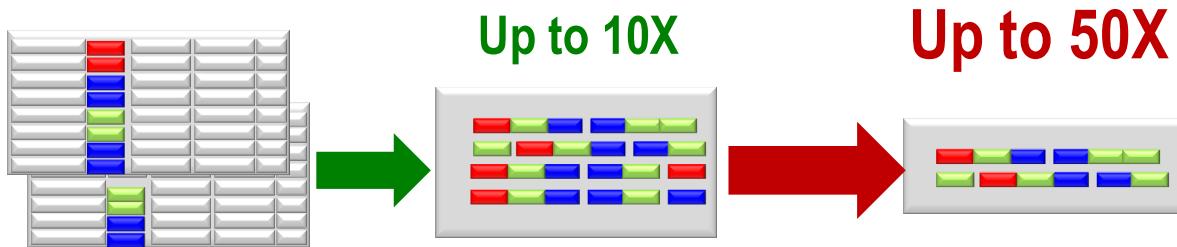
- Storage tiering
- OLTP / HCC compression for Archive

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

Oracle Database 11g In-Database Archiving can be achieved with three different solutions:

- **Partitioning** data to distinguish active data from those in archived state. ILM can therefore compress or move historical partitions to other low cost storage tiers.
- **Flashback Data Archive (FDA)** – Total Recall option in Oracle Database 11g that provides the ability to automatically track and archive transactional data changes to specified database objects. An FDA consists of multiple tablespaces storing historic data from all transactions against tracked tables for the time duration specified using the RETENTION parameter in partitioned compressed system tables. Historical data can be queried using the Flashback Query AS OF clause. Archived historic data that has aged beyond the retention period is automatically purged.
- **ILM Online Archive**, a storage tier, where all the data that is seldom accessed or modified are stored. The storage tier is likely to be extremely large and to store the maximum quantity of data. Various techniques can compress the data. This tier can be located in the database, stored on low-cost storage devices. The data is still online and available, for a cost that is only slightly higher than storing this information on tape, without the disadvantages that come with archiving data to tape. If the Online Archive storage tier is identified as read-only, then subsequent backups are not required after the initial database backup.

In-Database Archiving: HCC



- **Query mode** for data warehousing tables
 - Typical 10x compression ratios
- **Archival mode** for old data
 - Typical 15-50x compression ratios
- Data stored by column and then compressed

ORACLE

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

Oracle has implemented deeper table compression features, called Hybrid Columnar Compression, or HCC. It is designed for “In-Database archiving” providing 15-times to 50-times compression, the goal being to keep as much data as possible online in Oracle Database. This type of compression requires Exadata, ZFS, or Pillar Axiom 600 storage and has two main levels.

- **Warehouse Compression:** Is ideal for compressing data that are used for queries in a data-warehousing environment in which tables are being times compression ratio is achieved, meaning a 10-times reduction in the amount of I/O needed to complete a scan of a table, with a corresponding improvement in query performance.
- **Archive Compression:** Is designed for “In-Database Archiving” keeping as much data as possible online in Oracle Databases using a deeper compression ratio, and achieving up to 50 times compression ratios. It is best suited for data that is non-active, typically stored offline. But now Archive Compression delivers an online archive.

How does it work?

Typical database compression algorithms compress repeating values found within rows of data stored in a database. So for example, all the order dates within an order row will be compressed. However, as the rows are stored on disk in row format, there is a lot of non-relevant information stored between each occurrence of the next value of an order date : between each order date, additional values are found for order ID, customer ID, product IDs, and so on.

Hybrid Columnar Compression uses a different technique to store the column values. Instead of storing in a row format, the data is effectively stored by column: for instance, all the order dates will be stored together, then all the order IDs, then all the customer IDs, and so on. This means that within a unit of compression, a much higher rate of repeating values is found, and a greater compression ratio can be achieved. Each compression unit can span multiple data blocks. The values for a particular column may or may not span multiple blocks. The compression is no longer bound to one data block.

Archiving Challenges and Solutions

Challenges:

- Keep non-active and active data within the same table
- Access to active data only
- Decrease archived storage amount
- Keep performance access to archived data

Solutions:

- **In-Database Archiving**
 - Active
 - Not active
- **Temporal Validity**
 - Define valid-time dimension.
 - Filter on valid-time columns to access active data only.



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

Until now, archived data is not distinguishable from active data at **SEGMENT** level.

Oracle Database 12c provides two capabilities to distinguish active rows from those in non-active state within the same table:

- In-Database Archiving capability with row-archival management
- Temporal Validity: a new feature of Temporal

Note: The Total Recall option in Oracle Database 11g (the feature name is Flashback Data Archive known as FDA) is now called “Temporal” in Oracle Database 12c.

Temporal includes:

- The existing feature Flashback Data Archive is now called “Temporal History”
- A new feature “Temporal Validity”.

In-Database Archiving, enhanced in Oracle Database 12c with row-archival management, indicates as a row property the activeness of row data. Distinct lifecycle states for rows in a table can be defined as active and not active.

Temporal Validity provides the ability to add a valid-time dimension consisting of two date-time columns specified in the table definition. Using a valid-time dimension for each row of a table is a way of indicating operational relevance. If the data is no longer valid, then this data can be archived and hidden in regular queries. Processing a small active data set on a table that grows in size becomes more efficient. Data visibility relies on an explicit or implicit predicate on the valid-time dimension. While achieving operational efficiency of archiving, the applications easily access only data that is in an operationally active state although archived data is kept in the same table.

In-Database Archiving

- Enable row-archival: ORA_ARCHIVE_STATE column added.

```
SQL> CREATE TABLE emp
  2      (EMPNO NUMBER(7), FULLNAME VARCHAR2(40),
  3       JOB VARCHAR2(9), MGR NUMBER(7))
  4   ROW ARCHIVAL;
```

- New rows: ORA_ARCHIVE_STATE value 0
- Update ORA_ARCHIVE_STATE to 1 for row archiving.



ORACLE

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

To distinguish active rows from those in non-active state within the same table, and then compress the latter ones, use the new In-Database Archiving capability.

The row-archival attribute indicates as a row property the activeness of row data.

Distinct states for a row in a table can be defined: active, non-active.

To set the value of the activeness of a row data, ROW ARCHIVAL needs to be enabled at the segment level. This adds a new column ORA_ARCHIVE_STATE that will contain the state of each row.

By default, the row once inserted is active, and the ORA_ARCHIVE_STATE value is '0'.

After a certain elapsed time, the row can be less frequently accessed and rarely updated, but still considered as active. Then the row will be rarely accessed and no more updated, it is retained for records retention and/or compliance purposes: it is considered in a non-active state. The value '1' - or any value other than '0' - in the ORA_ARCHIVE_STATE column would reflect the non-active state. The two possible values of ORA_ARCHIVE_STATE are 0 or 1.

ORA_ARCHIVE_STATE column

- View ORA_ARCHIVE_STATE column.

```
SQL> SELECT ORA_ARCHIVE_STATE, fullname FROM emp;
```

ORA_ARCHIVE_STATE	FULLNAME
0	JEAN
1	ADAM
1	TOM
1	JIM

Active

Not active

Not active

Not active

- Set rows in archive state.

```
SQL> UPDATE emp
  2      SET ORA_ARCHIVE_STATE = DBMS_ILM.ARCHIVESTATENAME(1)
  3 WHERE empno < 100;
```

- Set rows back in active state.

```
SQL> UPDATE emp
  2      SET ORA_ARCHIVE_STATE = DBMS_ILM.ARCHIVESTATENAME(0);
```



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

Data in a non-active state can be compressed and be subject to storage compression.

As a hidden column, ORA_ARCHIVE_STATE is visible only when explicitly specified in select-lists of queries.

The administrator has to update the value of ORA_ARCHIVE_STATE column to one to set the row in archive state.

Session Visibility Control

- Session visibility control: by default, active rows only

```
SQL> select ORA_ARCHIVE_STATE, fullname from emp;
```

ORA_ARCHIVE_STATE	FULLNAME
	JEAN

JEAN ← Active

- Enable application visibility of all rows.

```
SQL> alter session set ROW ARCHIVAL VISIBILITY = ALL;
```

```
SQL> select ORA_ARCHIVE_STATE, fullname from emp;
```

ORA_ARCHIVE_STATE	FULLNAME
	JEAN

0	JEAN	←	Active
1	ADAM	←	Not active
1	JIM	←	Not active

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

By default, sessions have the visibility of active rows only.

The application visibility of active data can be enabled at the session-level. Rows that were moved to the non-active lifecycle state are filtered by the system, and only active rows are visible.

The ALTER SESSION statement is extended with a new ROW ARCHIVAL VISIBILITY clause. By default, the value is ALL. ACTIVE value allows viewing only active data, and not non-active data, in the session.

Disable Row-Archival

- Disable row-archival: ORA_ARCHIVE_STATE column removed

```
SQL> ALTER TABLE emp NO ROW ARCHIVAL;
```



ORACLE

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

When you disable the row-archival attribute on a table because there is no need anymore to distinguish active and non-active rows, the ORA_ARCHIVE_STATE column is automatically dropped.

Quiz

Which statement is true about row-archival?

- a. The ORA_ARCHIVE_STATE column always exists at table creation.
- b. Oracle automatically sets appropriate values in the ORA_ARCHIVE_STATE column when data ages.
- c. By default, the ORA_ARCHIVE_STATE column is set to 0 when a new row is inserted.
- d. The ORA_ARCHIVE_STATE column is part of the Temporal Validity feature.



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

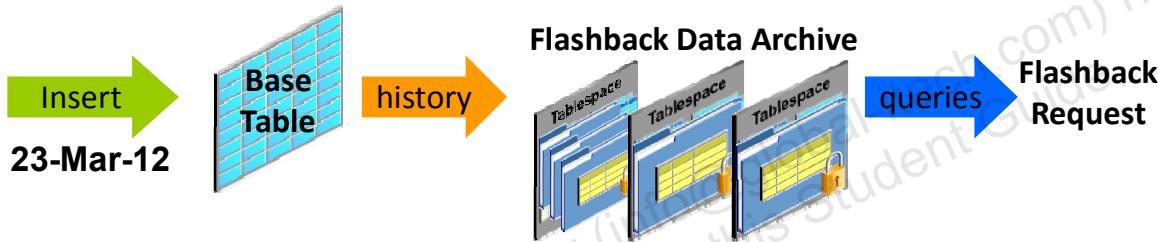
Answer: c

Temporal Validity Versus Temporal History

- Temporal Validity

Emp IDc	Job	Hire Date
100	Clerk	22-Apr-11
200	Developer	12-Dec-11
400	Salesman	22-Mar-12

- Temporal History



ORACLE

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

With the In-Database Archiving, you can distinguish active rows from non-active rows within the same table, and then compress the non-active rows. You now look at another way to achieve the same goal with Temporal Validity.

Applications often annotate the validity of a fact recorded in a table with dates or timestamps that are relevant to the underlying business; for example, the hire date of an employee in HR applications, or the effective date of coverage in the insurance industry. This temporal attribute is called temporal validity, and is usually controlled by the user or application who defines the valid-time dimension at table creation.

Temporal validity dates or time stamps are different than the dates or timestamps indicating when the fact was recorded in the database. The date or timestamp when the fact was recorded in the database are attributes of Temporal History (also known as Flashback Data Archive) and are system-managed. The Oracle Database 11g Total Recall feature (or FDA) implements the transaction time of temporal history.

In the slide, employee 400 was hired on Mar 22, but the row was entered in the HR.EMP table on Mar 23. Mar 22 is the valid time temporal start date and Mar 23 is the transaction time temporal start date.

By using the valid time temporal implicit filter on the valid-time dimension, queries can show rows that are currently valid or that will be valid in the future. The query is able to hide rows whose facts are not currently valid.

Bi-temporal queries can use both valid time temporal and transaction time temporal date.

PERIOD FOR Clause Concept

- Keep both active and non-active data in the same table.
- Define one valid-time dimension at table creation.
 - Explicitly define the two date-time columns.

```
SQL> CREATE TABLE emp
  2      ( empno number, salary number, deptid number,
  3           name VARCHAR2(100),
  4           user_time_start DATE, user_time_end DATE,
  5 PERIOD FOR user_time (user_time_start,user_time_end));
```

- Or implicit automatic valid-time columns are created.

```
SQL> CREATE TABLE emp2
  2      ( empno number, salary number, deptid number,
  3           name VARCHAR2(100),
  3 PERIOD FOR user_time);
```

- Insert rows explicitly naming the valid-time columns.



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

A valid-time dimension, represented by the new `PERIOD FOR` clause, consists of two date-time columns that can be specified in the table definition, as shown in the first example or that are automatically created, as shown in the second example.

To hide valid-time dimension columns, just specify a `PERIOD FOR` clause name without any date columns. Oracle creates two hidden columns using the name of the valid-time dimension as a prefix for the names of the two columns. The valid-time dimension name is used to drop the dimension if required. As shown in the second example, you defined the `user_time` as the name of the valid-time dimension and `user_time` is used as the prefix for the two date columns automatically created `USER_TIME_START` and `USER_TIME_END`.

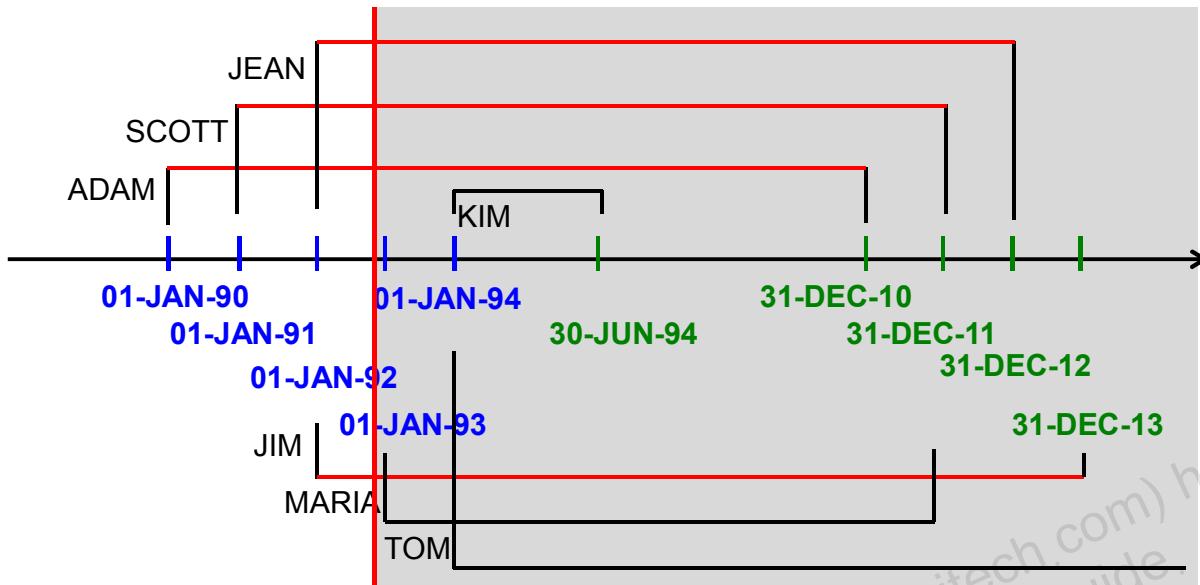
To insert rows into a table with a valid-time dimension, you would name the two valid-time date columns as follows:

```
SQL> INSERT INTO emp2 (empno, salary, deptid, name,
  2                      user_time_start, user_time_end)
  3 VALUES (1,1000,20, 'John', SYSDATE, NULL);
SQL> select EMPNO, user_time_start, user_time_end from emp2;
   EMPNO USER_TIME_START          USER_TIME_END
-----
```

1 17-AUG-12 09.58.03.000000 AM +00:00

Filtering on Valid-Time Columns: Example 1

Filter on valid-time columns to access active data only



```
SQL> select * from hr.emp as of PERIOD FOR user_time
2          to_date('01-DEC-1992', 'dd-mon-yyyy') ;
```

ORACLE

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

How do you filter on valid-time columns? Use the `SELECT` statement with the `PERIOD FOR` clause or use the `DBMS_FLASHBACK_ARCHIVE` procedure.

- There is one set of data that is “valid” based on its valid-start and valid-end times and the query time (AS OF or undecorated).
- On the other hand there is the other set of rows where the query time falls outside the valid-start and valid-end times.

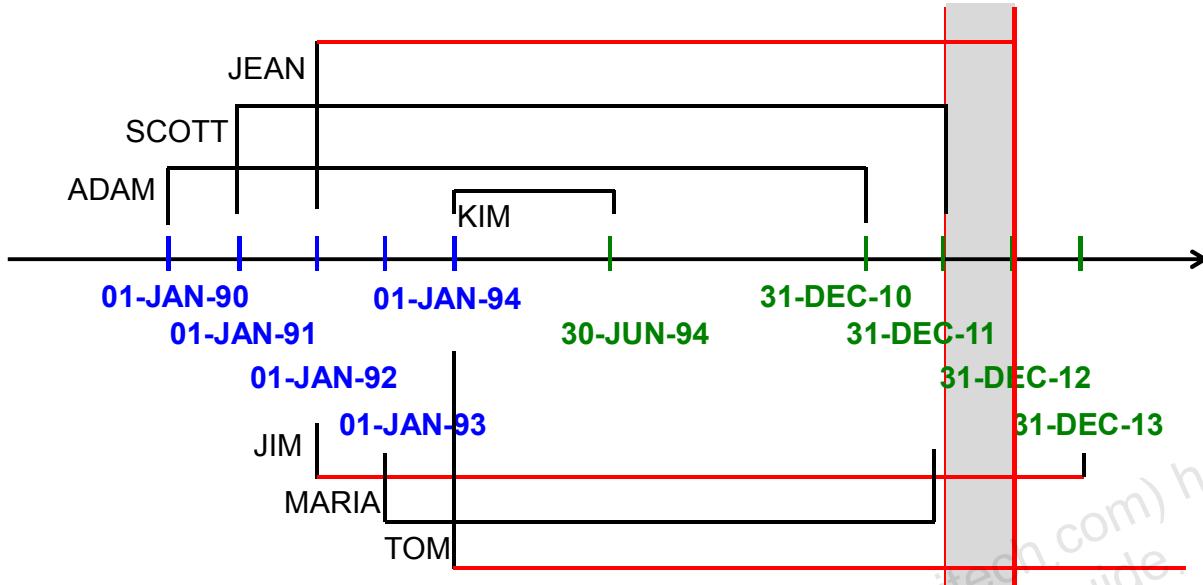
Both sets of rows data reside in the same table. However by controlling the visibility of data to the valid rows, you can limit what queries and DMLs affect. Until now, you could do as-of and versions queries for transaction time. Now you can do as-of and versions queries for valid time.

For each new employee that you inserted in the table, you included the hire dates, valid-time start dates, and valid end dates. The dates represent the activeness of each row. These dates are entered by the application and correspond to valid dates. The time that the rows were inserted and committed in the table corresponds to the transaction date.

You can filter the active employees by using the following new `PERIOD FOR` clause. The query displays all active employees who were valid at the explicit date of '01-DEC-1992', which is the date that belongs to the valid period; that is, between `USER_TIME_START` and `USER_TIME_END`.

Filtering on Valid-Time Columns: Example 2

Use versions queries for valid-time



```
SQL> SELECT * FROM hr.emp VERSIONS PERIOD FOR user_time
  2  BETWEEN to_date('31-DEC-2011','DD-MON-YYYY')
  3  AND    to_date('31-DEC-2012','DD-MON-YYYY');
```

ORACLE

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

You can filter the active employees by using the following new VERSIONS PERIOD FOR BETWEEN clause:

```
SQL> select * from hr.emp VERSIONS PERIOD FOR user_time
  2  BETWEEN to_date('31-DEC-2011', 'dd-mon-yyyy')
  3  AND    to_date('31-DEC-2012', 'dd-mon-yyyy');
```

The query displays all employees whose VALID_TIME_START is less than or equal to '31-DEC-2011' and VALID_TIME_END greater than or equal to '31-DEC-2012'.

Queries that mix valid-time and transaction-time dimensions are called bi-temporal queries.

This example shows rows as of the specified transaction time, which are valid now.

```
SQL> select * from hr.emp
  2  as of period for user_time to_date('31-DEC-1992', 'dd-mon-yyyy')
  3  as of timestamp to_date ('30-mar-2012','dd-mon-yyyy');
```

DBMS_FLASHBACK_ARCHIVE

- Visibility control applies to queries and DML.
- Full visibility applies to DDL.
- Visibility set with DBMS_FLASHBACK_ARCHIVE:
 - Set the visibility to data valid as of the given time

```
DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('ASOF',
                                             (to_timestamp('29-SEP-10 05.44.01 PM')))
```

- Set the visibility to data currently valid within the valid time period at the session level.

```
DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('CURRENT')
```

- Set the visibility to data to the full table level.

```
DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('ALL')
```



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

Users can modify visibility within a session via new DBMS_FLASHBACK_ARCHIVE package. Visibility control applies to all SQL SELECT and DML statements.

DDLS will default to getting full visibility to the table data. For example, CTAS, online redefinition and ALTER TABLE MOVE operations will have full visibility of the table data. Hidden columns are also visible to the DDL operations, resulting in preservation of those columns and their data.

The first example sets the valid time visibility to '29-SEP-2010', showing only rows overlapping the given date.

The second example sets the visibility of to data currently valid within the period time at the session-level.

The third example sets the visibility to the full table which is the default temporal table visibility.

Quiz

Choose the correct statement regarding the Temporal Validity feature.

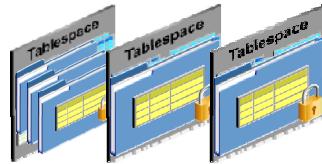
- a. Temporal Validity and temporal history cannot be used in the same query.
- b. Temporal Validity visibility control can be set at session-level.
- c. Temporal Validity columns are automatically updated by Oracle.



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

Answer: b

Temporal History Enhancements: FDA Optimization

**Pre-12c**

History or archive tables:

- Compressed and deduplicated

12c

History or archive tables:

- **Not** compressed or deduplicated

```
SQL> CREATE FLASHBACK ARCHIVE fla1 TABLESPACE tbs1
      RETENTION 5 YEAR;
```

12c

History or archive tables: compressed or deduplicated

```
SQL> CREATE FLASHBACK ARCHIVE fla1 TABLESPACE tbs1
      OPTIMIZE DATA RETENTION 5 YEAR;
```

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

Flashback Data Archive in Oracle Database 11g

In Oracle Database 11g, FDA history tables are compressed, deduplicated and internally partitioned.

Temporal History in Oracle Database 12c

In Oracle Database 12c, FDA history tables are no more compressed nor deduplicated at FDA creation. Creating a table with the `OPTIMIZE DATA` clause, table and LOB compression and LOB deduplication is automatically turned on, using any of the following features: Advanced Row table compression, Advanced LOB Compression, Advanced LOB Deduplication, segment-level and row-level ADO compression. ADO is enabled to allow new data to be archived as uncompressed and over time is compressed in the background.

FDA history tables already compressed and deduplicated in 11g or before upgrade to 12c will continue to get compression and deduplication.

To stop optimization on FDA history tables, execute the following statement:

```
SQL> ALTER FLASHBACK ARCHIVE fla1 NO OPTIMIZE DATA;
```

Temporal History Enhancements: User Context Metadata

The user context of transactions on FDA enabled tables is:

- Collected by an attribute: NONE (default), TYPICAL, ALL
 - Set with DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL
 - If TYPICAL, collects database user id, global user id, client identifier, service name, module name or host name
- Obtained from USERENV using GET_SYS_CONTEXT

```
SQL> EXEC DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL('TYPICAL')
PL/SQL procedure successfully completed
```

```
SQL> SELECT DBMS_FLASHBACK_ARCHIVE.GET_SYS_CONTEXT
  2      (VERSIONS_XID, 'USERENV','SESSION_USER'),
  3      VERSIONS_XID,VERSIONS_STARTTIME,VERSIONS_ENDTIME,
  4      employee_id, salary
  5  FROM hr.employees VERSIONS BETWEEN SCN MINVALUE
  6                                AND MAXVALUE ;
```

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

In Oracle Database 11g, Flashback Data Archive does not collect the user context data of transactions executed on FDA enabled tables. Oracle Database 12c allows now the retrieval of this information.

- The user context of a transaction executed on a Flashback Data Archive enabled table is collected and retrievable. The parameters of the namespace USERENV describe the current session. The user context is obtained from DBMS_FLASHBACK_ARCHIVE.GET_SYS_CONTEXT.
- The user context collection is controlled by a parameter, set by DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL whose values can be NONE, TYPICAL or ALL. By default no user context is collected. Some examples of information, which are collected in the TYPICAL case, are database user ID, global user ID, client identifier, service name, module name or host name.
- Rows in these tables are purged when the commit time is older than the retention of the flashback archive with the longest retention.
- Each row of the user context can only be read by the DBA or the owner of the transaction.

Summary

In this lesson, you should have learned how to:

- Distinguish In-Database Archiving from Temporal Validity
- Enable In-Database Archiving for visibility of active data only
- Describe the advantages of temporal validity versus temporal history
- Set temporal validity at table level
- Use a new SQL temporal data type
- Use temporal validity to reflect business validity
- Understand session-level or flashback point-in-time query
- Describe Temporal History (FDA) enhancements



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

Practice 8 Overview: In-Database Archiving and Temporal

These practices cover the following topics:

- Using In-Database Archiving with Row-archival
- Using Temporal Validity to automatically view valid-time rows only
- Using user context information in the history table of FDA-enabled tables (*optional*)

Module Security



ORACLE

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

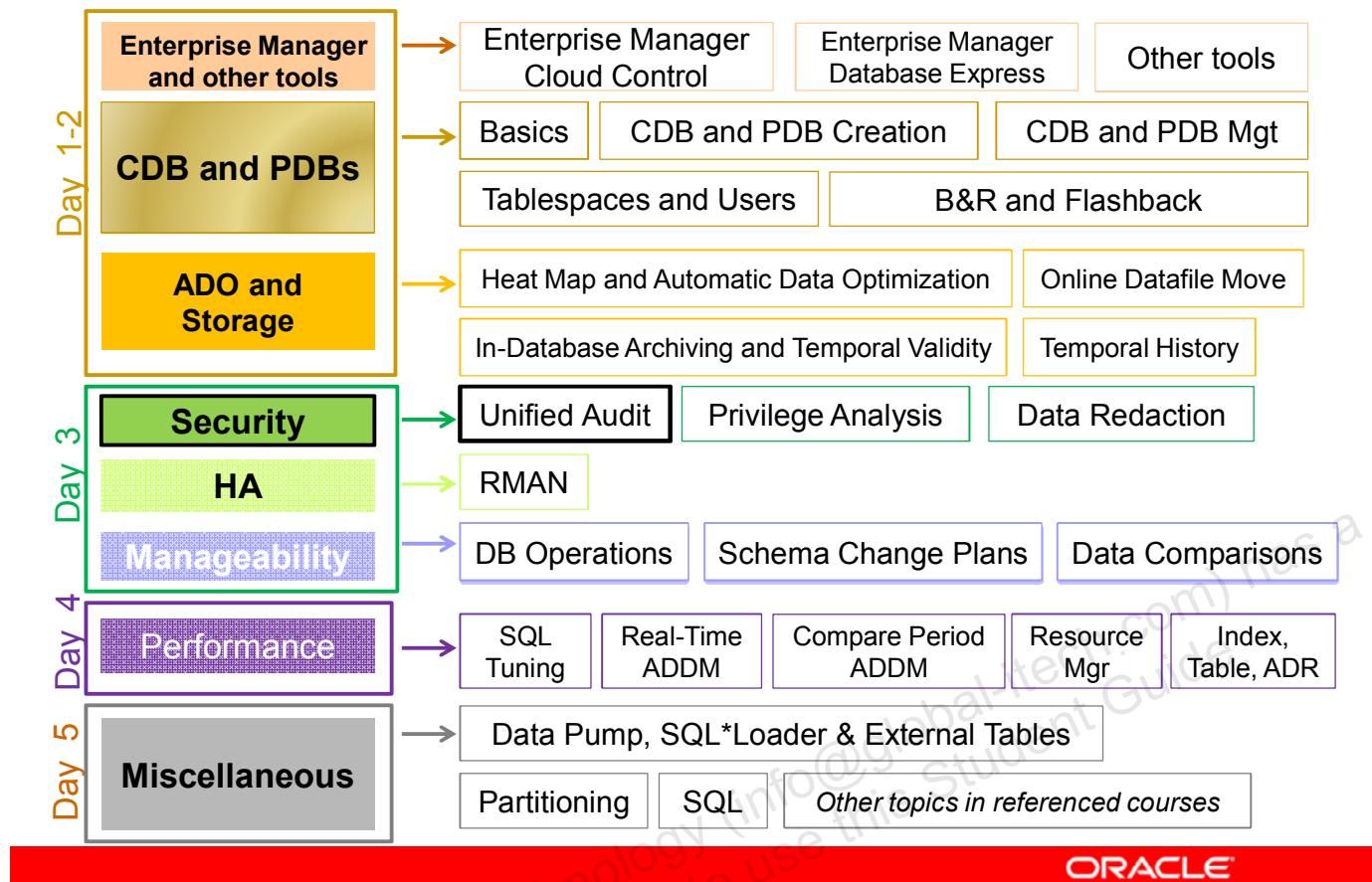
9

Auditing

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.

Unified Audit: The unified audit facility serves to consolidate all audit trails into a single unified audit trail table, improve audit performance, secure audit data for all RDBMS and other components like RMAN and Data Pump, and simplify audit configuration.

Objectives

After completing this lesson, you should be able to:

- Understand the implementation and benefits of using the unified audit trail
- Enable unified auditing
- Configure the unified auditing
- Create and enable audit policies
- Add application context data to audit records
- View data in the unified audit trail
- Clean up the unified audit trail



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

Note

For a complete understanding of Oracle Unified Auditing new feature and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Security Guide 12c Release 1 (12.1)*

Refer to other sources of information:

- *Oracle Database 12c New Features Demo Series* demonstrations under Oracle Learning Library:
 - *Unified Auditing*
- *Oracle By Example (OBE)*:
 - *Auditing With Unified Auditing*

Types of Auditing

Monitor activity:

Type of Auditing	Activity Audited
Mandatory auditing	<ul style="list-style-type: none"> • STARTUP, SHUTDOWN • SYSDBA, SYSOPER, SYSASM connections
Standard database auditing	AUDIT_TRAIL parameter
Privileged users auditing	AUDIT_SYS_OPERATIONS = TRUE
Fine-grained auditing	DBMS_FGA.ADD_POLICY
Oracle Database Vault auditing	Any violation against any component: <ul style="list-style-type: none"> • Realm, Command Rule, Secure Application Role • Rule, Factor



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

Monitoring for Activity

Compliance, privacy, and increasingly sophisticated security threats have resulted in an almost mandatory requirement for turning on native auditing. This is especially true for monitoring privileged users, but also for actions that may be out of the normal operations that one would expect to see in a production environment. In fact, privileged users have increasingly been the target of external hackers. As a result, auditing privileged user activity and application changes such as DDL operations are especially important. Database link creation, create table, and truncate table are just a few of the DDL operations that should be audited. Starting with Oracle Database 11g, auditing for these types of operations is configured by default.

Oracle Database 11g provides three different types of auditing. The administrator can audit all actions that take place within the database. It is important to remember that auditing should be focused so that only events that are of interest are captured. Properly focused auditing has minimal impact on system performance and provide highly relevant data for reporting and alerting purposes.

- Standard database auditing captures several pieces of information about an audited event, including that the event occurred, when it occurred, the user who caused the audited event, and which client machine was being used when the event happened.
- Audit privileged user operations with the AUDIT_SYS_OPERATIONS parameter.
- Independently of standard database auditing, use fine-grained auditing to audit SQL data manipulation language statements occurring only when specific columns are impacted under certain conditions.

Audit Trail Implementation

Store audit records:

Audit Trail	Activity Audited
Send audit records to DB / OS files	AUDIT_TRAIL = DB / OS / XML DB, EXTENDED XML, EXTENDED AUDIT_FILE_DEST = directory_path
Move SYS.AUD\$, SYS.FGA_LOG\$ tables to defined tablespace	Use DBMS_AUDIT_MGMT package SET_AUDIT_TRAIL_LOCATION procedure
DVSYS.AUDIT_TRAIL\$	No other possibility for the Oracle Database Vault audit

Secure audit data:

- syslog audit: AUDIT_SYSLOG_LEVEL=local1.info
- Oracle Audit Vault



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

Storing Audit Records

The **AUDIT_TRAIL** parameter defines where audit records will be written. This parameter is set to DB by default. If **AUDIT_TRAIL** is set to a value of NONE, standard database auditing is disabled. The **AUDIT_TRAIL** parameter is a static parameter and cannot be modified without restarting the database instance.

The possible parameter values that enable auditing are:

- **DB**: Enables database auditing and directs all audit records to the database audit trail (the **SYS.AUD\$** table)
- **DB, EXTENDED**: In addition to action of the DB value, captures the SQL executed along with any bind variables (**SQLTEXT** and **SQLBIND** columns) of the audit table
- **XML**: Enables database auditing and writes all audit records to the XML format OS files
- **XML, EXTENDED**: Similar behavior as DB, EXTENDED
- **os**: Enables database auditing and directs audit records to the operating system files.

For all **AUDIT_TRAIL** options that write to the OS files, the default directory is determined by **AUDIT_FILE_DEST**.

Oracle Database Vault audit data is stored in the **DVSYS.AUDIT_TRAIL\$** table.

Compliance regulations and laws require businesses to secure business and personal data for customers, employees, and partners. Businesses must demonstrate compliance with the regulations by auditing users, activities, and associated data.

- AUDIT_TRAIL can be set to OS, and the audit files are created in the AUDIT_FILE_DEST directory and are owned by the oracle user.
- If you also set the AUDIT_SYSLOG_LEVEL parameter to values, the audit files are owned by root, and they are visible only to root. The records are sent to the syslog process.
- A more secure solution is Oracle Audit Vault, which is able to:
 - Transparently collect and consolidate audit data from several Oracle databases
 - Help detect and prevent insider threats by alerting you to suspicious activity
 - Help organizations simplify compliance reporting with built-in and custom reports

Oracle Database 12c Auditing

- **Simplicity:**
 - Audit options grouped into a simple audit policy
 - Simpler action-based audit configuration
 - Condition-based audit configuration
 - Users exempted from being audited
- **Consolidation:** One single unified audit trail
- **Security:**
 - Read-only audit trail table
 - Any operation related to audit configuration audited
 - Any SYS user actions audited
 - Separation of audit administration duties with audit roles
- **Performance:**
 - Negligible overhead

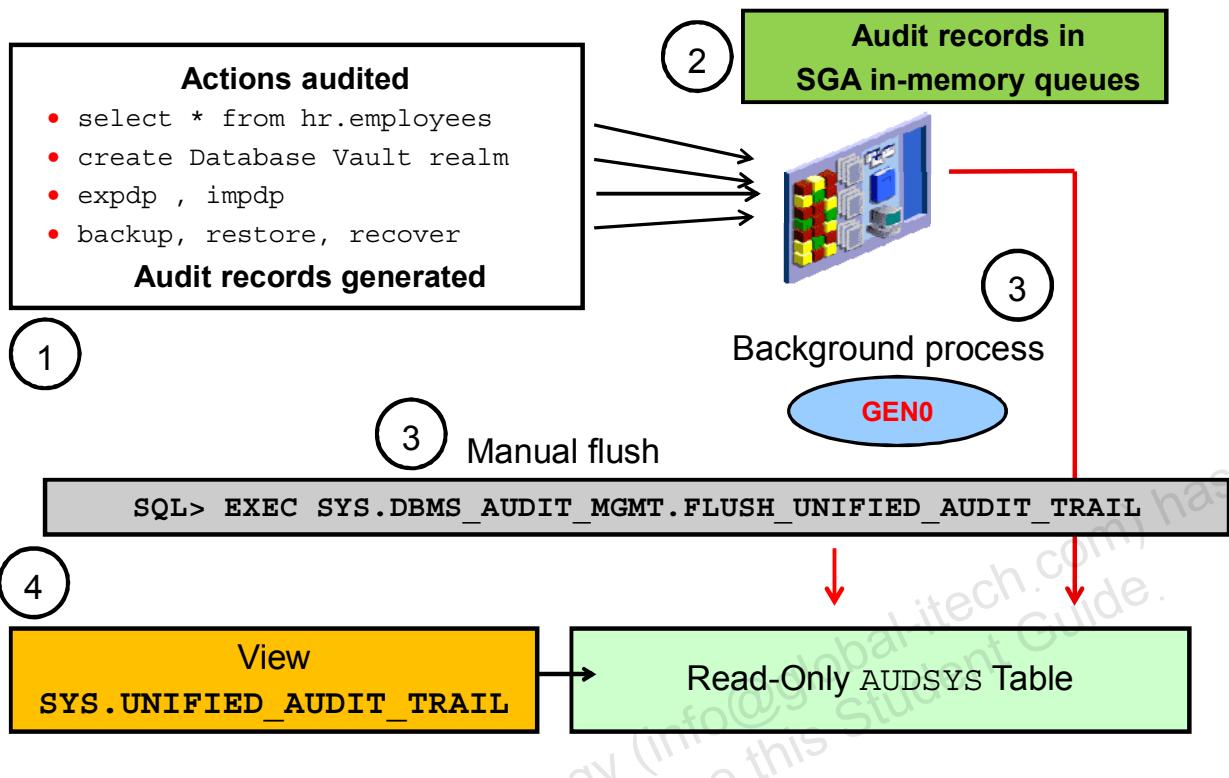


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The unified auditing facility needs to address the following challenges:

- **Simplicity:**
 - Grouping audit options into a simple audit policy
 - Allowing simpler action-based audit configurations
 - Setting condition-based audit configurations
 - Exempting users from being audited
- **Consolidation:** Merging all audit trails into a single unified audit trail table
- **Security:**
 - Relying on a read-only audit trail table
 - Auditing any operation related to audit configuration
 - Auditing any SYS user auditable action
 - Separating audit administration duties with audit administration roles, AUDIT_ADMIN and AUDIT_VIEWER
- **Performance:**
 - Oracle Database 12c auditing provides the ability to use System Global Area (SGA) queues for accumulating audit records, enabling auditing to be turned on with negligible overhead.

Security and Performance: Audit Architecture



ORACLE

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

How does the new audit architecture work?

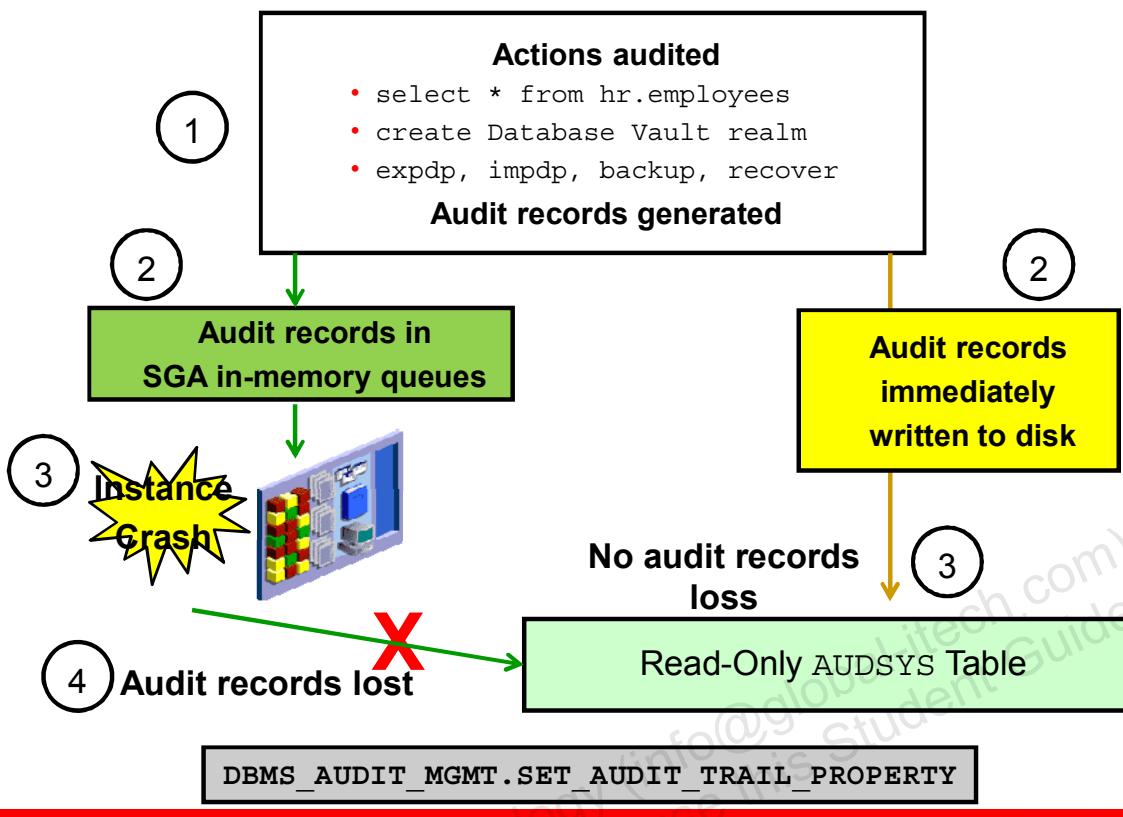
Named queues inside the SGA have a corresponding persistent storage in tables owned by AUDSYS.

When the SGA queue overflows, a background process flushes the contents to the table. Each client has two SGA queues so that a client can continue to write to the second queue while the first queue is being persisted to the table.

When you want to see the audit trail records immediately in the `UNIFIED_AUDIT_TRAIL` view, use the following procedure to explicitly flush the queues to disk:

```
SQL> EXEC SYS.DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL
```

Tolerance Level for Loss of Audit Records



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

ORACLE

The in-memory queuing infrastructure means that the audit records are not persisted immediately but queued, thus achieving the performance benefit. However, in cases like instance crash, queued audit records can be lost. Hence, the audit trail must be configurable to indicate the tolerance level. The following two modes are based on different levels of loss tolerance:

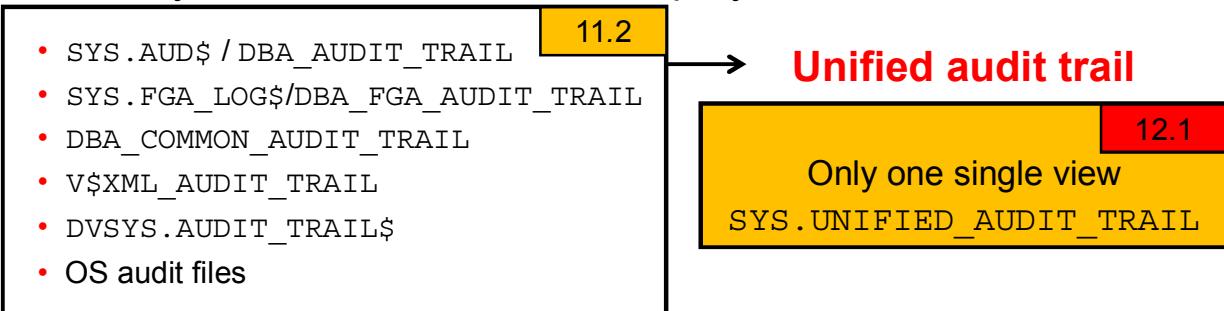
- Immediate-Write mode: The audit records are written immediately. This comes with a performance cost.

```
SQL> EXEC DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY (
  2   DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED, -
  3   DBMS_AUDIT_MGMT.AUDIT_TRAIL_WRITE_MODE, -
  4   DBMS_AUDIT_MGMT.AUDIT_TRAIL_IMMEDIATE_WRITE);
```
- Queued-Write mode: The content of the SGA queues is periodically dumped to disk. This is the default mode for the new audit engine.

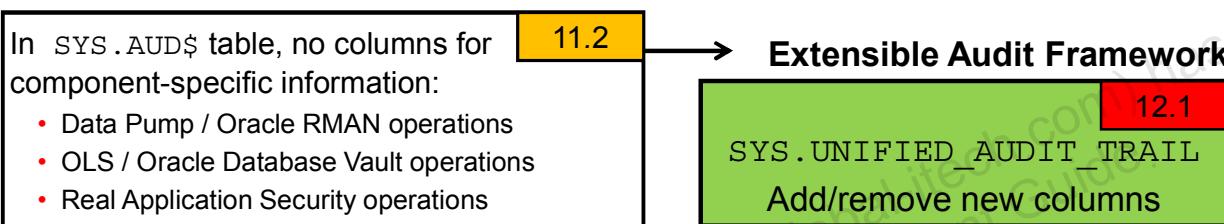
```
SQL> EXEC DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_PROPERTY (
  2   DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED, -
  3   DBMS_AUDIT_MGMT.AUDIT_TRAIL_WRITE_MODE, -
  4   DBMS_AUDIT_MGMT.AUDIT_TRAIL_QUEUED_WRITE);
```

Consolidation: Unique Audit Trail

Too many audit trails to be looked up by auditors:



No easy extension to add an audit column to the audit trail:



Management and security of audit trail improved with a single audit trail.

ORACLE

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

- In Oracle Database 11g, the Oracle database has too many audit trails where the audited data must be looked up by auditors. The first major challenge is to consolidate all audit trails in a unified audit trail.
- The existing audit trail tables do not easily extend to add or remove new audit columns for newer RDBMS components, like Data Pump, Oracle Label Security (OLS), or Oracle Database Vault, to supply additional component-specific information. The second challenge is therefore to allow components to extend the basic audit information via the extensible audit framework to supply their own additional audit information in the unified audit trail. For example, in Oracle Database 11g, OLS actually stores its audit data in the `SYS.AUD$` table, but there are no specific columns related to OLS specific information.
- The management and security of audit trail becomes easier with a single audit trail.

Basic Audit Versus Extended Audit Information

Basic Audit Information record

- Database session:
- User name, Database Client
 - Terminal, IP Address
 - Instance number, DBID
- Database operation:
- Action executed
 - SCN
 - Object accessed, SQL statement

Basic Audit Information

BAI in view
SYS.UNIFIED_AUDIT_TRAIL

Extended Audit Information columns

- For component-specific information:
- FGA: `FGA_POLICY_NAME`
 - Data Pump operations: `DP_xxx`
 - RMAN operations: `RMAN_xxx`
 - OLS operations: `OLS_xxx`
 - DV violations/changes: `DV_xxx`
 - RAS operations: `XS_xxx`

Extended Audit Information

EAI in view
SYS.UNIFIED_AUDIT_TRAIL
New columns

ORACLE

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

Basic Audit Information

In Oracle Database 11g, the attributes combining Database Session information and Database Operation information constitute the basic format of an audit record in Oracle Database. This basic format is called Basic Audit Information or BAI for short.

The SQL AUDIT command configures the audit for SQL operations and connections.

Extended Audit Information

Apart from these attributes, the different RDBMS components could have their own actions, like the Data Pump “Direct Path Load” actions or the OLS “Apply Policy” or the Oracle Database Vault “Create Command Rule” which are PL/SQL procedures. Any database component, like FGA, Data Pump, Oracle RMAN, OLS, Oracle Database Vault (DV), or Real Application Security (RAS), can supply additional component-specific information and store it in the unified audit trail view. This additional component-specific information is called Extended Audit Information, or EAI for short.

Though the audit trail is unified, the audit records appear separately for each type of auditing. For example, if the audit is configured in both standard auditing and Oracle Database Vault, separate audit records corresponding to each of these components appear in the audit trail.

Extended Audit Information

EAI in SYS.UNIFIED_AUDIT_TRAIL view: new columns

FGA event	<ul style="list-style-type: none"> • FGA_POLICY_NAME
Data Pump Export / Import	<ul style="list-style-type: none"> • DP_TEXT_PARAMETERS1 • DP_BOOLEAN_PARAMETERS1
RMAN backup / recovery	<ul style="list-style-type: none"> • RMAN_OPERATION • RMAN_OBJECT_TYPE • RMAN_DEVICE_TYPE • RMAN_xxx
OLS operations	<ul style="list-style-type: none"> • OLS_POLICY_NAME • OLS_xxx
Database Vault violations or configuration changes	<ul style="list-style-type: none"> • DV_xxx
Real Application Security	<ul style="list-style-type: none"> • XS_xxx

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

EAI in the UNIFIED_AUDIT_TRAIL New Columns

A database component can be audited like:

- FGA component, which is registered with the framework. The EAI is stored in the FGA_POLICY_NAME column.
- Data Pump when export or import operations are executed. The EAI is stored in two new columns: DP_TEXT_PARAMETERS1 and DP_BOOLEAN_PARAMETERS1.
- Oracle RMAN when backup or restore or recovery operations are executed. The EAI is stored in several new columns: RMAN_OPERATION would contain either 'Backup' or 'Restore' or 'Recover'.
- OLS when applying an OLS policy on a table, for example. The EAI is stored in the OLS_xxx columns.
- Oracle Database Vault when configuration changes or violations occur against Oracle Database Vault components, such as realms, command rules, and rulesets factors. The EAI is stored in the DV_xxx columns.
- RAS operations. The EAI is stored in the XS_xxx columns.

Data Pump Audit Policy

1. Create the audit policy for the component and actions.

```
SQL> CREATE AUDIT POLICY dp_pol1 ACTIONS
      2          COMPONENT=datapump export;
```

2. Enable the audit policy.

```
SQL> audit policy DP_POL1;
```

3. Perform an export operation.

```
$ expdp hr/oracle_4U dumpfile=HR_tables tables=EMPLOYEES,JOBS
      directory=DATA_PUMP_DIR
```

4. Flush the audit data to disk.

```
SQL> EXEC SYS.DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL
```

5. View the Data Pump EAI.



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

Data Pump EAI in UNIFIED_AUDIT_TRAIL New Columns

If you want the Data Pump Export or Import operations to be audited, you perform these steps:

1. Create an audit policy for export or import or for both. An audit policy is a named group of audit settings that enable you to audit a particular aspect of user behavior in the database.
2. Next, enable the audit policy that was created to audit export operations.
3. Perform an export operation. At each export operation, the EAI is then stored in two new columns of the UNIFIED_AUDIT_TRAIL view: DP_TEXT_PARAMETERS1 and DP_BOOLEAN_PARAMETERS1.
4. Flush the audit information to disk by using the DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL procedure.
5. Display the Data Pump audit information in the UNIFIED_AUDIT_TRAIL view.

Oracle RMAN Audit Information

1. Perform an Oracle RMAN operation.

```
RMAN> backup tablespace users;  
RMAN> restore tablespace users;  
RMAN> recover tablespace users;
```

2. Flush the audit data to disk.

```
SQL> EXEC SYS.DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL
```

3. View the Oracle RMAN EAI.

```
SQL> select DBUSERNAME, RMAN_OPERATION from UNIFIED_AUDIT_TRAIL  
  2 where RMAN_OPERATION is not null;  
  
DBUSERNAME          RMAN_OPERATION  
-----  
SYSBACKUP           Backup  
SYSBACKUP           Restore  
SYSBACKUP           Recover
```



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

RMAN EAI in the UNIFIED_AUDIT_TRAIL New Columns

Unlike other Oracle Database components, you do not create a unified audit policy for Oracle RMAN events. The UNIFIED_AUDIT_TRAIL view automatically captures commonly audited Oracle RMAN events.

Unified Audit Implementation

→ Mixed auditing mode

Allow smooth migration of existing databases to use the unified auditing features.

→ Unified auditing mode

Before populating the UNIFIED_AUDIT_TRAIL view:

- Enable unified auditing.
 1. Shut down all processes and database instances of the ORACLE_HOME.
 2. cd \$ORACLE_HOME/rdbms/lib
make -f ins_rdbms.mk **uniaud_on** ioracle
 3. Start all Oracle processes of all instances.
- Define a tablespace for the read-only audit table.

The AUDIT_xxx instance parameters are no longer observed.



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

Mixed Auditing Mode

Mixed auditing mode allows both the new audit engine and the traditional audit engine to work simultaneously.

- On upgrade, none of the existing audit settings are affected and the database continues to support the traditional audit facilities. It is recommended you migrate to the unified audit facility to take advantage of the new audit architecture and performance improvements. Before you decide to switch to the unified auditing mode, you can use the mixed mode by creating a policy with CREATE AUDIT POLICY command and then enabling it with AUDIT command. If you do not wish to create a new policy, you can simply enable one of the predefined policies - ORA_SECURECONFIG or ORA_ACCOUNT_MGMT or ORA_DATABASE_PARAMETER. Either of this puts the database in mixed auditing mode. The old audit syntax continues to work and the old audit destinations continue to be written to. However, new audit policies can be created and audit data also writes to the new unified audit trail.
- When a database is created, mixed auditing mode is used by default through the predefined enabled policy ORA_SECURECONFIG. But unified auditing mode is not yet enabled.

Unified Auditing Mode

To enable unified auditing, relink Oracle with the `uniaud_on` option after shutting down all Oracle processes and database instances in the `ORACLE_HOME`. By default, the new unified audit data is written to the `SYSAUX` tablespace. If you do not want to use the `SYSAUX` tablespace, create a new tablespace to host the new read-only audit trail table, which is owned by the new `AUDSYS` schema. Note that the `AUDSYS` user is locked by default. After enabling unified audit mode, none of the old audit instance parameters are honored: `AUDIT_TRAIL`, `AUDIT_FILE_DEST`, `AUDIT_SYS_OPERATIONS`, `AUDIT_SYSLOG_LEVEL`. There is no possibility to write audit records to OS or syslog.

Quiz

To audit RMAN backup, restore, and recovery operations, you must create new audit policies.

- a. True
- b. False



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

Answer: b

Unlike other Oracle Database components, you do not create a unified audit policy for Oracle RMAN events. The `UNIFIED_AUDIT_TRAIL` view automatically captures commonly audited Oracle RMAN events.

Quiz

Select the schema owner of the unified audit trail.

- a. SYS
- b. AUDSYS
- c. AUDIT_ADMIN
- d. LBACSYS
- e. DVSYS

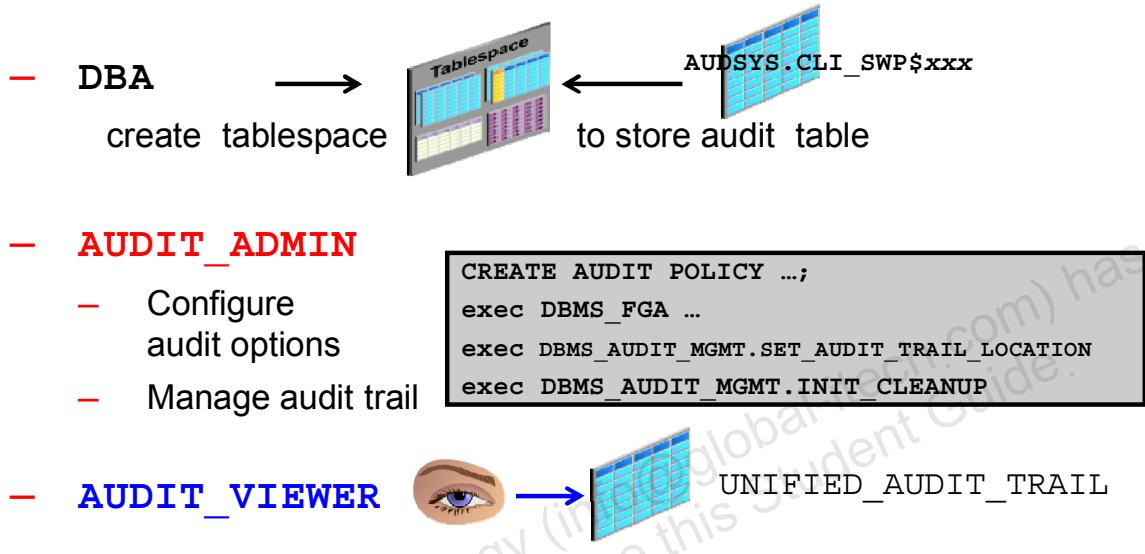


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

Security: Roles

- Read-only audit trail table in AUDSYS schema
- Any audit-related activity mandatorily audited
- Roles to administer auditing and view audit data:



ORACLE

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

Addressing Oracle Database 12.1 Security Challenges

Compliance regulations and laws require businesses to secure audit data by auditing users, activities, and associated data.

Read-Only Audit Trail Table

The audit trail table owned by the AUDSYS schema is in read-only mode, and can be updated only during recovery or upgrade operations by using the SYSDBA privilege.

Audit Configuration Operations

Any audit-related activity, such as creation, modification, deletion, enabling of audit policies, and execution of DBMS_AUDIT_MGMT and DBMS_FGA packages, are audited by mandate.

Roles

The SYSDBA privilege is no longer required to configure audit settings and management of the audit trail. To enable separation of duties, the AUDIT_ADMIN role is required to perform these functions with the granted privileges:

- AUDIT SYSTEM: Create, alter, drop audit policies
- AUDIT ANY: Enable, disable audit policies by using the AUDIT command
- SELECT privilege on various audit views and EXECUTE privilege on DBMS_AUDIT_MGMT and DBMS_FGA packages

The AUDIT_VIEWER role allows for viewing and analyzing the audit data.

Security: SYS Auditing

Non Unified Auditing

```
AUDIT_SYS_OPERATIONS=true  
AUDIT_FILE_DEST=directory
```

SYSDBA, SYSOPER, SYSASM, SYSBACKUP,
SYSKM, SYSDG:
Statements recorded:
STARTUP, SHUTDOWN, ALTER DATABASE
+
SELECT, DML, DDL

OS audit directory

Unified Auditing

```
AUDIT_SYS_OPERATIONS=true  
AUDIT_FILE_DEST=directory
```

SYSDBA, SYSOPER, SYSASM, SYSBACKUP,
SYSKM, SYSDG:
Statements recorded:
STARTUP, SHUTDOWN, ALTER DATABASE ...
+
Enabled actions in audit policies

SYS.UNIFIED_AUDIT_TRAIL

ORACLE

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

When SYS is connected as SYSDBA, SYSOPER, SYSBACKUP, SYSASM, SYSKM, or SYSDG, it is subjected to:

- All top-level statements, such as STARTUP, SHUTDOWN, ALTER DATABASE, and ALTER SYSTEM, until the database opens.
- System-wide audit policies configured by the audit administrator. When the database opens, Oracle Database audits these users by using the audit configurations in the system—not only the ones that were applied in the AUDIT statement for SYS. It audits all users when the AUDIT statement does not define a list of users for which the policy applies.
- Audit when a table or a PL/SQL procedure is configured for auditing.

The audit administrator can configure audit policies that apply to the SYS user. This configuration is covered later in the lesson.

Simplicity: Audit Policy

- No more AUDIT_xxx instance parameters
- To start auditing, perform the following steps:
 1. Create audit policies used to group audit options:
CREATE AUDIT POLICY command:
 - Audit options: System-wide or object-specific or role
 - Optional **WHEN** condition **EVALUATE PER**
STATEMENT
 - Optional event-handler alert
 - **CONTAINER** = CURRENT | ALL
 2. Enable/disable audit policies: AUDIT and NOAUDIT
 - Define users audited: all by default
 - User list: BY username
 - User exception list: EXCEPT username
- View audited data in **UNIFIED_AUDIT_TRAIL** view



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

In a unified audit database, the following actions are audited by mandate: CREATE AUDIT POLICY, ALTER AUDIT POLICY, DROP AUDIT POLICY, AUDIT, NOAUDIT, EXECUTE of DBMS_FGA, and EXECUTE of DBMS_AUDIT_MGMT.

Auditing actions based on system or object privileges or roles are performed, provided that you create and enable audit policies. What are audit policies?

- A named group of audit options that enable you to audit a particular aspect of user behavior in the database by using system, object, or role privileges
- Created with the CREATE AUDIT POLICY command requiring the AUDIT_ADMIN role
- Enforced conditionally and evaluated during the statement execution or once during the session or at the instance level
- Supplied with an event-handler procedure alerting that the audit occurred
- Applied in a multitenant container database (CDB) either in the root or a pluggable database (PDB)
- Not active until explicitly enabled with the AUDIT command
Be aware that existing standard non-policy audit options configured with the AUDIT and NOAUDIT commands are not effective.
- Applied with the exception of certain user lists.

Step 1: Creating the Audit Policy

Create audit policies based on **system-wide** audit options.

- System privilege

```
SQL> CREATE AUDIT POLICY audit_syspriv_pol1
  2      PRIVILEGES SELECT ANY TABLE, CREATE LIBRARY;
```

- Actions

```
SQL> CREATE AUDIT POLICY audit_actions_pol2
  2      ACTIONS AUDIT, ALTER TRIGGER;
```

- Role

```
SQL> CREATE AUDIT POLICY audit_role_pol3
  2      ROLES mgr_role;
```

- System privilege, actions, and roles

```
SQL> CREATE AUDIT POLICY audit_mixed_pol4
  2      PRIVILEGES DROP ANY TABLE
  3      ACTIONS     CREATE TABLE, DROP TABLE, TRUNCATE TABLE
  4      ROLES       emp_role;
```



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

Audit Policy Creation: System-Wide Options

To create an audit policy, you must first define system-wide or object-specific audit options.

The system-wide options can be of three types:

- The **privilege audit option** audits all events, which exercise the specified system privilege, as in the first example on the slide.
- The **action audit option** indicates which RDBMS action should be audited, such as the ALTER TRIGGER action in the second example.
- The **role audit option** audits the use of all system or object privileges granted directly to the role MGR_ROLE, as in the third example.

You can configure privilege, action, and role audit options in the same audit policy, as shown in the fourth example.

Find the list of auditable system-wide options in the SYS.AUDITABLE_SYSTEM_ACTIONS table.

Creating the Audit Policy: Object-Specific Actions

Create audit policies based on **object-specific** options.

```
SQL> CREATE AUDIT POLICY audit_objpriv_pol5
  2      ACTIONS SELECT, UPDATE,
  3                  LOCK ON hr.employees;
```

```
SQL> CREATE AUDIT POLICY audit_objpriv_pol6
  2      ACTIONS ALL;
```

```
SQL> CREATE AUDIT POLICY audit_objpriv_pol7
  2      ACTIONS EXECUTE,
  3                  GRANT ON hr.raise_salary_proc;
```



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

Object-specific options are the second type of audit option. These options are actions that are specific to objects in the database. The object-level action audit options are listed in the `SYS.AUDITABLE_OBJECT_ACTIONS` table.

The first example on the slide creates an audit policy to audit any select and update action on any object, and any lock on the `HR.EMPLOYEES` table.

The second example creates an audit policy to audit any object-specific action on any object.

The third example creates an audit policy to audit any execute action on any procedural object, and any grant on the `HR.RAISE_SALARY_PROC` procedure.

The object-level audit options are dynamic. That is, changes in these options become applicable for current and subsequent user sessions.

Creating the Audit Policy: Condition

- Condition and evaluation PER SESSION

```
SQL> CREATE AUDIT POLICY audit_mixed_pol5
  2      ACTIONS RENAME ON hr.employees,
  3                  ALTER ON hr.jobs,
  4 WHEN 'SYS_CONTEXT (''USERENV'', ''SESSION_USER'')='JIM'
  5 EVALUATE PER SESSION;
```

- Condition and evaluation PER STATEMENT

```
SQL> CREATE AUDIT POLICY audit_objpriv_pol6
  2      ACTIONS ALTER ON OE.ORDERS
  3 WHEN
  4   'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'')='OE'
  5 EVALUATE PER STATEMENT;
```

- Condition and evaluation PER INSTANCE

```
SQL> CREATE AUDIT POLICY audit_objpriv_pol7
  2      ROLES dba
  3 WHEN 'SYS_CONTEXT (''USERENV'', ''INSTANCE_NAME'')='orcl'
  4 EVALUATE PER INSTANCE;
```



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

Audit Policy Creation: Condition and Evaluation

The audit policies can evaluate the condition per statement, once per session or once per instance.

The first example on the slide creates an audit policy to audit any rename action on the HR.EMPLOYEES table, and any alter action on the HR.JOBs table, provided that the user executing the audited statement is JIM. The condition is evaluated only once in the session.

The second example creates an audit policy to audit any alter action on the OE.ORDERS table, provided that the user executing the audited statement is the schema owner OE. The condition is evaluated each time an ALTER statement is executed on the OE.ORDERS table.

The third example creates an audit policy to audit any privilege granted to the DBA role, provided that the instance name is "orcl". The condition is evaluated only once during the database instance lifetime. After Oracle Database evaluates the condition, it caches and reuses the result for the remainder of the instance lifetime.

Step 2: Enabling / Disabling the Audit Policy

Enable audit policies:

- Apply to all users.

```
SQL> AUDIT POLICY audit_syspriv_pol1;
```

- Apply only to some users.

```
SQL> AUDIT POLICY audit_pol2 BY scott, oe;
SQL> AUDIT POLICY audit_pol3 BY sys;
```

- Exclude some users.

```
SQL> AUDIT POLICY audit_pol4 EXCEPT jim, george;
```

- Audit the recording based on failed or succeeded actions.

```
SQL> AUDIT POLICY audit_syspriv_pol1 WHENEVER SUCCESSFUL ;
SQL> AUDIT POLICY audit_objpriv_pol2 WHENEVER NOT SUCCESSFUL ;
```

```
SQL> AUDIT POLICY auditpol5 BY joe WHENEVER SUCCESSFUL ;
```

Disable audit policies by using the NOAUDIT command.



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

Audit Policy Enabling

The next step is to enable the audit policy by using the AUDIT statement.

All users are audited by default, except if you define a list of those audited.

- Apply the audit policy to one or more users by using the BY clause.
The audit Administrator audits SYS-user-specific actions the same way as other user actions.
- Exclude some users by using the EXCEPT clause.
You cannot have a BY list and an EXCEPT list in the same policy enablement statement

Audit records are generated whether the user's actions failed or succeeded. If you want to audit actions only when the user's actions failed, use the WHENEVER NOT SUCCESSFUL clause. If you want to audit actions only when the user's actions succeeded, use the WHENEVER SUCCESSFUL clause. When you omit the WHENEVER clause, the statement is audited whether the action is successful or not, and the RETURN_CODE column displays whether the action succeeded or not.

Audit Policy Disabling

To disable an audit policy, use the NOAUDIT command.

Viewing the Audit Policy

- View audit policy.

```
SQL> SELECT POLICY_NAME, AUDIT_OPTION, CONDITION_EVAL_OPT
  2  FROM  AUDIT_UNIFIED_POLICIES;
```

POLICY_NAME	AUDIT_OPTION	CONDITION_EVAL_OPT
POL1	DELETE	INSTANCE
POL2	TRUNCATE TABLE	NONE
POL3	RENAME	SESSION
POL4	ALL ACTIONS	STATEMENT

- View the enabled audit policy.

```
SQL> SELECT POLICY_NAME, ENABLED_OPT, USER_NAME, SUCCESS, FAILURE
  2  FROM  AUDIT_UNIFIED_ENABLED_POLICIES;
```

POLICY_NAME	ENABLED_OPT	USER_NAME	SUC	FAI
POL3	BY	PM	NO	YES
POL2	EXCEPT	SYSTEM	NO	YES
POL4	BY	SYS	YES	YES
POL6	BY	ALL USERS	YES	NO

ORACLE

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

Verify the list of audit policies created in the AUDIT_UNIFIED_POLICIES view.

- The CONDITION_EVAL_OPT column defines when the condition, if any, is evaluated.

Verify the list of enabled audit policies in the AUDIT_UNIFIED_ENABLED_POLICIES view.

- The ENABLED_OPT column defines if a list of audited users is defined with the BY value or an exception list of excluded users is defined with the EXCEPT value. The audited or excluded users are listed in the USER_NAME column.
- The SUCCESS and FAILURE columns define if the policy generates audit records only when the user's actions succeed or fail.

Using Predefined Audit Policies

```
SQL> select POLICY_NAME, AUDIT_OPTION
  2  from  AUDIT_UNIFIED_POLICIES
  3 where policy_name in
  4       ('ORA_ACCOUNT_MGMT',
  5        'ORA_DATABASE_PARAMETER',
  6        'ORA_SECURECONFIG')
  7  order by 1,2 ;

POLICY_NAME          AUDIT_OPTION
-----              -----
ORA_ACCOUNT_MGMT      ALTER ROLE
ORA_ACCOUNT_MGMT      ALTER USER ...
ORA_DATABASE_PARAMETER ALTER DATABASE
ORA_DATABASE_PARAMETER ALTER SYSTEM
ORA_DATABASE_PARAMETER CREATE SPFILE
ORA_SECURECONFIG        ADMINISTER KEY MANAGEMENT ...
```



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

There are three main predefined audit policies for operations, which are commonly audited and recommended under Oracle's Audit Best Practices.

- The ORA_ACCOUNT_MGMT audit policy audits events for the user account and privilege management.
- The ORA_DATABASE_PARAMETER audit policy audits the changes made to the database parameters settings.
- The ORA_SECURECONFIG audit policy audits all Oracle Database 11g secure configuration audit options.

The users must enable or disable such precreated audit policies, based on their requirements.

Only the ORA_SECURECONFIG is enabled by default.

```
SQL> select POLICY_NAME from AUDIT_UNIFIED_enabled_policies
  2  where policy_name in ('ORA_ACCOUNT_MGMT',
  3        'ORA_DATABASE_PARAMETER', 'ORA_SECURECONFIG');

POLICY_NAME
-----
ORA_SECURECONFIG
```

Including Application Context Data

Context Namespace	Attribute / Value
CONTEXT_HR	MyJob – Developer MyLoc – Paris
CONTEXT_SALES	MyRole – Manager

Use the AUDIT CONTEXT ATTRIBUTES command.

```
SQL> AUDIT CONTEXT NAMESPACE context_HR
  2       ATTRIBUTES MyJob, MyLoc
  3   CONTEXT NAMESPACE context_SALES
  4       ATTRIBUTES MyRole BY jim;

SQL> select APPLICATION_CONTEXTS from UNIFIED_AUDIT_TRAIL
  2 where APPLICATION_CONTEXTS is not null;
APPLICATION_CONTEXTS
-----
(context_HR,MyJob=Manager) ; (context_HR,MyLoc=Paris)
```

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

Capturing Application Context Information in the Unified Audit Trail

Applications set various contexts before executing actions on the database server, storing information like Module Name and Client_ID. These values are typically stored in application contexts. An application context is an attribute name and attribute value pairs in a namespace, and each pair can be appended in audit data.

The audit administrator can configure application contexts and their attributes that must be captured in audit data by using the AUDIT CONTEXT NAMESPACE ATTRIBUTE command.

To display the audited application context data, select from the UNIFIED_AUDIT_TRAIL view.

To list the application contexts audited, use the AUDIT_UNIFIED_CONTEXTS view:

```
SQL> select * from audit_unified_contexts;
NAMESPAC          ATTRIBUTE          USER_NAME
-----            -----            -----
CONTEXT_HR        MYJOB             ALL_USERS
CONTEXT_HR        MYJOB             ALL_USERS
CONTEXT_SALES     MYROLE            JIM
```

To disable an application context audit setting, execute the NOAUDIT statement.

Dropping the Audit Policy

1. Disable the audit policy.

```
SQL> NOAUDIT POLICY audit_syspriv_pol1;
```

2. Drop the audit policy.

```
SQL> DROP AUDIT POLICY audit_syspriv_pol1;
```



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

To drop a unified audit policy, first disable it, and then run the `DROP AUDIT POLICY` statement to remove it. If a unified system-related audit policy is already enabled for a session, the effect of dropping a system-related audit policy is not seen by this existing session. Until that time, the unified system-related audit policy's settings remain in effect. However, the effect is immediate for object-related unified audit policies.

In a CDB environment, you can drop a common audit policy only from `root` and a local audit policy only from the PDB to which it applies.

Audit Cleanup

Clean up records in SYS.UNIFIED_AUDIT_TRAIL:

- Schedule an automatic purge job.

```
DBMS_AUDIT_MGMT.CREATE_PURGE_JOB
(AUDIT_TRAIL_TYPE=> DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
AUDIT_TRAIL_PURGE_INTERVAL => 12,
AUDIT_TRAIL_PURGE_NAME => 'Audit_Trail_PJ',
USE_LAST_ARCH_TIMESTAMP => TRUE,
CONTAINER => DBMS_AUDIT_MGMT.CONTAINER_CURRENT);
```

- Manually purge the audit records.

```
DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(
AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED)
```

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

To perform the audit trail purge tasks, you use the DBMS_AUDIT_MGMT package. You must have the AUDIT_ADMIN role to use the package. By mandate, Oracle Database audits all executions of the DBMS_AUDIT_MGMT package procedures.

The DBMS_AUDIT_MGMT.CREATE_PURGE_JOB procedure owns a new attribute CONTAINER. Setting CONTAINER to CONTAINER_CURRENT deletes audit trail records for the current pluggable database. Setting CONTAINER to CONTAINER_ALL deletes audit trail records for all pluggable databases creating a job in the root and the invocation of this job will invoke cleanup in all the PDBs.

Specifies whether the last archived time stamp should be used for deciding on the records that should be deleted.

Setting USE_LAST_ARCH_TIMESTAMP to TRUE indicates that only audit records created before the last archive time stamp should be deleted. A value of FALSE indicates that all audit records should be deleted. The default value is TRUE. Oracle recommends using this value, as this helps guard against inadvertent deletion of records.

You can automate the cleanup process by creating and scheduling a cleanup purge job, or you can manually run a cleanup purge job. If you manually run cleanup purge jobs, use the DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL procedure with a new type value of DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED.

Quiz

Select the view that you use to view the enabled audit policies.

- a. DBA_UNIFIED_AUDIT_POLICIES
- b. UNIFIED_AUDIT_ENABLED_POLICIES
- c. UNIFIED_AUDIT_POLICIES
- d. AUDIT_UNIFIED_ENABLED_POLICIES
- e. UNIFIED_AUDIT_TRAIL



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

Answer: d

Summary

In this lesson, you should have learned how to:

- Configure and enable unified audit
- Distinguish Basic Audit Information from Extended Audit Information
- Use the unified audit trail for auditing Data Pump and Oracle RMAN operations
- Configure the unified audit trail for auditing record loss tolerance
- Create and enable audit policies
- Add application context data to audit records
- View data in the unified audit trail
- Clean up the unified audit trail



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

Practice 9 Overview: Auditing

These practices cover the following topics:

- Demo: *Unified Auditing*
- Optional practice:
 - Enabling unified auditing mode
 - Creating audit policy for Data Pump export operations
 - Displaying audit records from `UNIFIED_AUDIT_TRAIL` view for RMAN operations

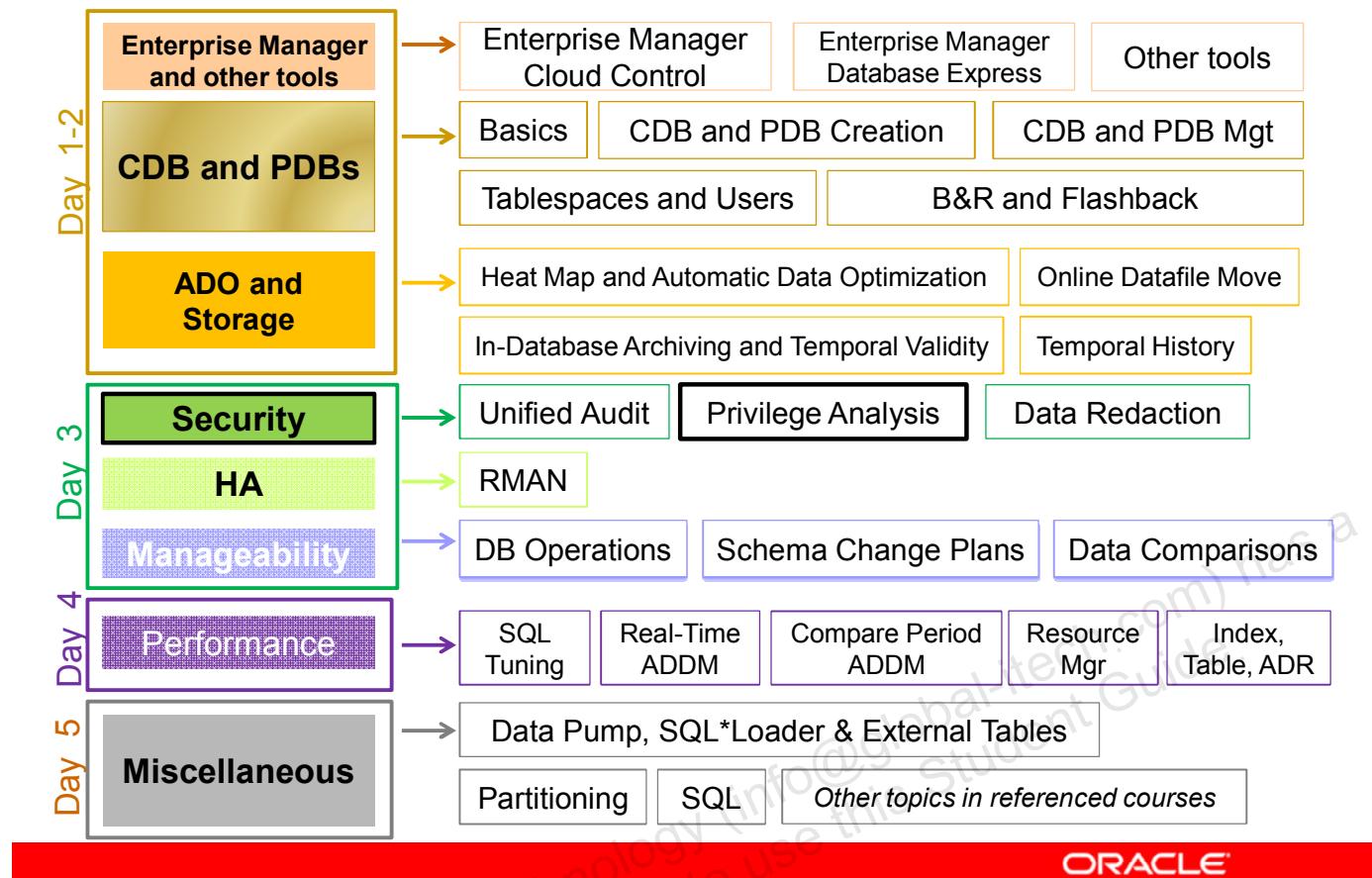
10

Privileges

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.

Privilege Analysis identifies unused privileges to be revoked.

Other privileges enhancements introduce new administrative system privileges which are more task-oriented to allow separation of duty.

Objectives

After completing this lesson, you should be able to:

- Implement separation of duty for database administration tasks
- Describe new administrative privileges
- Describe OS groups for administrative privileges
- Create the password file to support new privileges
- Explain the new system privilege PURGE DBA_RECYLEBIN
- Use database privilege analysis
- Create and enable privilege analysis
- Control privileges for invoker's rights procedures
- Manage the invoker's rights behavior for views



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

Note

For a complete understanding of the Privilege Analysis new feature and the new administrative privileges and other privileges usage, refer to the following guide in the Oracle documentation:

- *Oracle Database Security Guide 12c Release 1 (12.1)*
- *Oracle Database PL/SQL Packages and Types Reference 12c Release 1 (12.1) – “DBMS_PRIVILEGE_CAPTURE” chapter*

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

- *Oracle Database 12c New Features Demo Series* demonstrations :
 - *Administrative Privileges*
 - *Privileges Capture*
 - *INHERIT PRIVILEGES Privilege and Invokers' Rights Procedures*
 - *INHERIT PRIVILEGES Privilege and Current_User Views*
- *Oracle By Example (OBE)*:
 - *Determining Least Privilege Access Using Privilege Analysis*

Major Challenges

- **Separation of duty:** Administration tasks rely on SYSDBA.
 - New task-specific privileges for standard administrative tasks
 - Oracle RMAN administrator: backup and recovery
 - Data Guard administrator
 - Key management administrator: TDE keystore management
 - Security administrators set with Oracle Database Vault
 - DV owner and DV account manager
 - No change in SYSDBA privilege
 - Administrative privileges mandatorily audited
 - New system privilege: PURGE DBA_RECYLEBIN
- **Security:**
 - Identify unused privileges to revoke: Privilege Analysis
 - Check new privilege during invoker's rights procedures call
 - Check new privilege during invoker's rights views call



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

As one of the fundamental security requirements, the principle of separation of duty dictates that completion of any critical task has an associated role. In Oracle Database 11g, the administration of Oracle Database depends heavily on the SYSDBA administrative privilege.

For better separation of duty, Oracle Database now provides task-specific privileges to handle standard administration duties for Oracle Recovery Manager (Oracle RMAN), Oracle Data Guard, and transparent data encryption (TDE). The new privileges are based on the least privilege principle, in which a user is granted only the absolutely necessary privileges required to perform a specific function, and no more. This feature alleviates the need to unnecessarily grant the SYSDBA administrative privilege for many tasks. SYSDBA is necessary for critical situations, such as installation, upgrade, and urgent recovery.

The separation of duty improves database security by maintaining the privacy and the compliance requirements for data.

To achieve the least privilege principle, you must identify and revoke unnecessary privileges. Privilege Analysis helps identify used and unused privileges. The invoker's rights procedure calls let the invoker use all granted privileges during the procedure call. A new privilege gives invoking users control over who can access their privileges when they run an invoker's rights procedure. The same principle will apply to BEQUEATH CURRENT_USER views covered later in the lesson.

Administrative Privileges

Privilege	User	Authorized Operations
SYSDBA	SYS	STARTUP / SHUTDOWN ALTER DATABASE OPEN/MOUNT/BACKUP or change of character set CREATE / DROP DATABASE CREATE SPFILE ALTER DATABASE ARCHIVELOG / RECOVER Includes the RESTRICTED SESSION privilege Connected as user SYS
SYSOPER	PUBLIC	STARTUP / SHUTDOWN CREATE SPFILE ALTER DATABASE OPEN/MOUNT/BACKUP/ARCHIVELOG ALTER DATABASE RECOVER (Complete recovery) Includes the RESTRICTED SESSION privilege
SYSASM	SYS	ASM instance only, no access to database instance

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

The Oracle database management system (DBMS) currently provides two *administrative* privileges, SYSDBA and SYSOPER. The administrative privileges are “special” system privileges in that administrators who are granted these privileges can:

- Access a database instance even if the database is not open
- Perform basic administrative operations without having any other privileges.

Because of the unrestricted capability of the SYSDBA administrative privilege, you should grant the privilege only to highly trusted administrators because various administrative tasks and features require this privilege.

In the Oracle Database 11g release, the SYSASM administrative privilege was introduced for the administration of Automatic Storage Management (ASM) instances.

New Administrative Privileges

Administrative Privilege	Username	Tasks
SYSDBA, SYSOPER	SYS / PUBLIC	Same operations as in 11g
SYSASM	SYS	Specific to ASM instances only
SYSBACKUP	SYSBACKUP	Perform RMAN backup and recovery operations from RMAN or through SQL.
SYSDG	SYSDG	Perform Data Guard operations with Data Guard Broker or DGMGRL.
SYSKM	SYSKM	Manage transparent data encryption keystore operations.

- Distinction between administrative privileges (**SYSBACKUP**) and predefined users (**SYSBACKUP**) 
- **SYSBACKUP**, **SYSDG**, **SYSKM** users cannot be dropped.



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

Oracle Database 12c introduces new administrative privileges that are more task-specific and least privileged to support specific administrative tasks. The **SYSDBA** administrative privilege is not altered for backward compatibility. However, the connection with the **SYSDBA** administrative privilege should be limited to major operations such as installation and upgrade.

- The **SYSBACKUP** administrative privilege allows you to perform Oracle RMAN backup and recovery operations from Oracle RMAN or through SQL, STARTUP and SHUTDOWN and other operations.
- The **SYSDG** administrative privilege allows you to perform Data Guard operations with Data Guard Broker or the DGMGRL command-line interface, and STARTUP and SHUTDOWN and other operations.
- The **SYSKM** administrative privilege allows you to manage transparent data encryption keystore operations.

These privileges enable you to connect to the database even if the database is not open.

After connecting with one of these privileges, you are connected under a predefined user whose name is the privilege name. Each privilege is tied to a specific user.

```
SQL> connect / as SYSBACKUP
SQL> show user
USER is "SYSBACKUP"
```

New Administrative Privilege: SYSBACKUP

System / Object Privileges		
ALTER DATABASE ALTER SYSTEM CREATE SESSION ALTER SESSION ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE RESUMABLE	CREATE ANY DIRECTORY CREATE ANY TABLE CREATE ANY CLUSTER AUDIT ANY SELECT ANY DICTIONARY SELECT ANY TRANSACTION	SELECT X\$ tables, V\$ / GV\$ views EXECUTE SYS.DBMS_BACKUP_RESTORE SYS.DBMS_RCMAN SYS.DBMS_IR SYS.DBMS_TTS SYS.DBMS_TDB SYS.DBMS_PLUGTS SYS.DBMS_PLUGTSP
Statements and Roles		
CREATE PFILE CREATE SPFILE CREATE CONTROLFILE DROP DATABASE STARTUP , SHUTDOWN		CREATE / DROP RESTORE POINT (GUARANTEED restore points) FLASHBACK DATABASE SELECT_CATALOG_ROLE HS_ADMIN_SELECT_ROLE

- Connected as SYSBACKUP predefined user
- Can view tables' existence, but not application data



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

The SYSBACKUP administrative privilege allows the user being granted this privilege to:

- Perform STARTUP and SHUTDOWN operations
- Oracle RMAN tasks: backup, restore, recover (or connected to SQL*Plus) including TSPITR
- Create or drop a database, create a control file
- Perform ALTER DATABASE to change the ARCHIVELOG mode
- Flash back the database: create and drop guaranteed restore points
- Create an SPFILE or PFILE
- Alter the SYSAUX tablespace or drop it if the database is started in UPGRADE mode
- Audit any operation
- View the DBA_xxx, GV\$, and V\$ views but no SELECT on application tables

After connecting as AS SYSBACKUP, you are connected under the predefined SYSBACKUP user.

```
$ rman target ''/ as sysbackup"
RMAN> select user from dual;
USER
-----
SYSBACKUP
```

New Administrative Privilege: SYSDG

System / Object privileges	
CREATE SESSION ALTER SYSTEM ALTER SESSION ALTER DATABASE SELECT ANY DICTIONARY	SELECT X\$ tables, V\$ and GV\$ views DELETE / SELECT APPQOSSYS.WLM_CLASSIFIER_PLAN EXECUTE SYS.DBMS_DRS
Statements and Roles	
STARTUP SHUTDOWN	CREATE RESTORE POINT DROP RESTORE POINT (including GUARANTEED restore points) FLASHBACK DATABASE

- Connected as SYSDG predefined user
- Can view tables' existence, but not application data

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

The SYSDG administrative privilege allows the user being granted this privilege to:

- Perform STARTUP and SHUTDOWN operations
- Perform ALTER DATABASE to change the ARCHIVELOG mode
- Perform ALTER DATABASE RECOVER, including TSPITR
- Flash back the database
- Create and drop guaranteed restore points
- Start the observer
- Run DGMGRL
- Execute DBMS_DRS.INITIATE_FS_FAILOVER to allow an application to request the primary database to immediately invoke a fast-start failover
- Manage the primary and standby database instances
- Make trusted call-outs for LogMiner packages
- View the DBA_xxx, GV\$ and V\$ views but no SELECT on application tables

After connecting with the SYSDG privilege, you are connected under the predefined SYSDG user.

```
SQL> connect / as SYSDG
SQL> show user
USER is "SYSDG"
```

New Administrative Privilege: **SYSKM**

System / Object privileges

```

CREATE SESSION
ADMINISTER KEY MANAGEMENT
SELECT SYS.V$WALLET
SELECT SYS.V$ENCRYPTION_WALLET
SELECT SYS.V$ENCRYPTED_TABLESPACES

```

- Connected as **SYSKM** predefined user
- Manage TDE operations
 - Keystore creation, opening, closing
 - Master Key creation and changes
 - Column and tablespace keys management
 - Access to TDE information in appropriate views
- No access to application data



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

When tablespace encryption is in use, TDE encrypts data in REDO logs. Therefore, in case of recovery, encrypted REDO logs must be decrypted before they can be applied, and the Oracle keystore must be opened after the database is mounted and before the database is opened. Because of this requirement and because the **SYSDBA** administrative privilege is the only privilege that allows keystore operations during the mounted state, the encryption key management currently relies on the **SYSDBA** administrative privilege.

To maintain clear separation between database administration and encryption key management for compliance requirements, the **SYSKM** administrative privilege allows the user being granted this privilege to:

- Manage any TDE operation like keystore and keys management
- View the TDE-related views but no **SELECT** on application tables

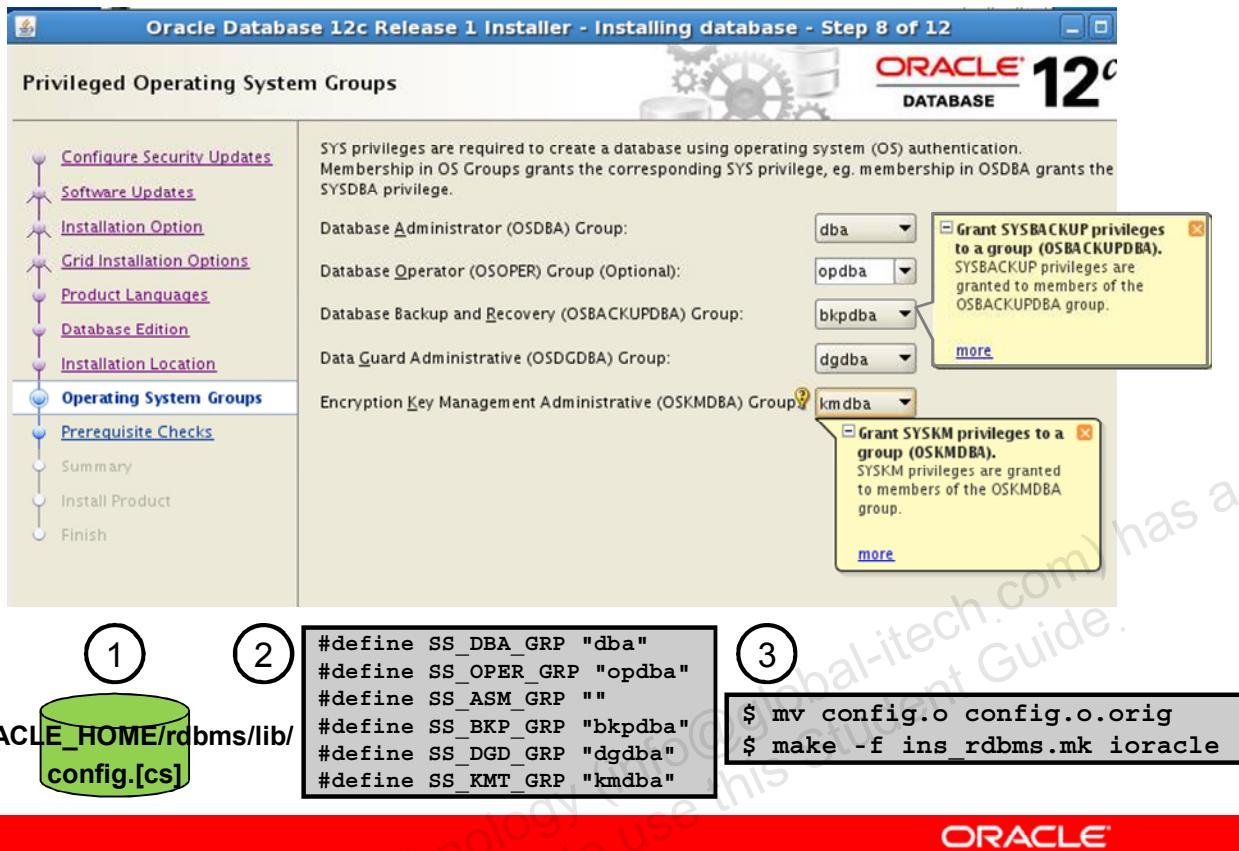
After connecting with the **SYSKM** privilege, you are connected under the predefined **SYSKM** user.

```

SQL> connect / as SYSKM
SQL> show user
USER is "SYSKM"

```

OS Authentication and OS Groups



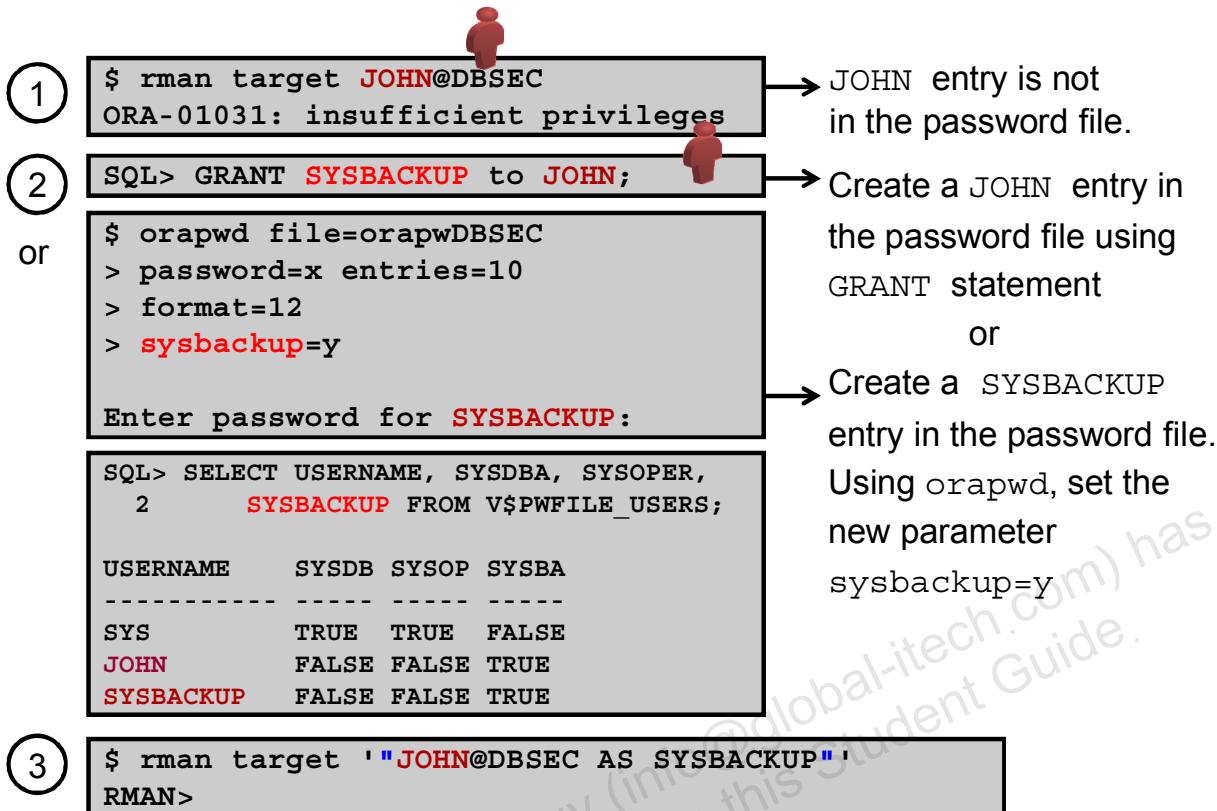
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Without being a member of these OS groups, users will not be able to connect as administrative users using the OS authentication. That is, “CONNECT / AS SYSDBA” will fail. However, the users can still connect using other authentication mechanisms (for example, network, password, or directory based authentication). To change the OS group names (as shown in step 1 and 2 on the slide), ensure that you are using the groups defined in the \$ORACLE_HOME/rdbms/lib/config.[cs] file. Next, shut down all databases, and then relink the Oracle executable as shown in step 3 on the slide.

Windows User Groups

The Windows user groups are ORA_%HOMENAME%_SYSBACKUP, ORA_%HOMENAME%_SYSDG, and ORA_%HOMENAME%_SYSKM. These user groups cannot be changed.

Password Authentication for SYSBACKUP



ORACLE

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

The password authentication still requires the `REMOTE_LOGIN_PASSWORDFILE` instance parameter set to `EXCLUSIVE` and a password file.

New parameters during the creation of the password file define new formats and supported new administrative privileges.

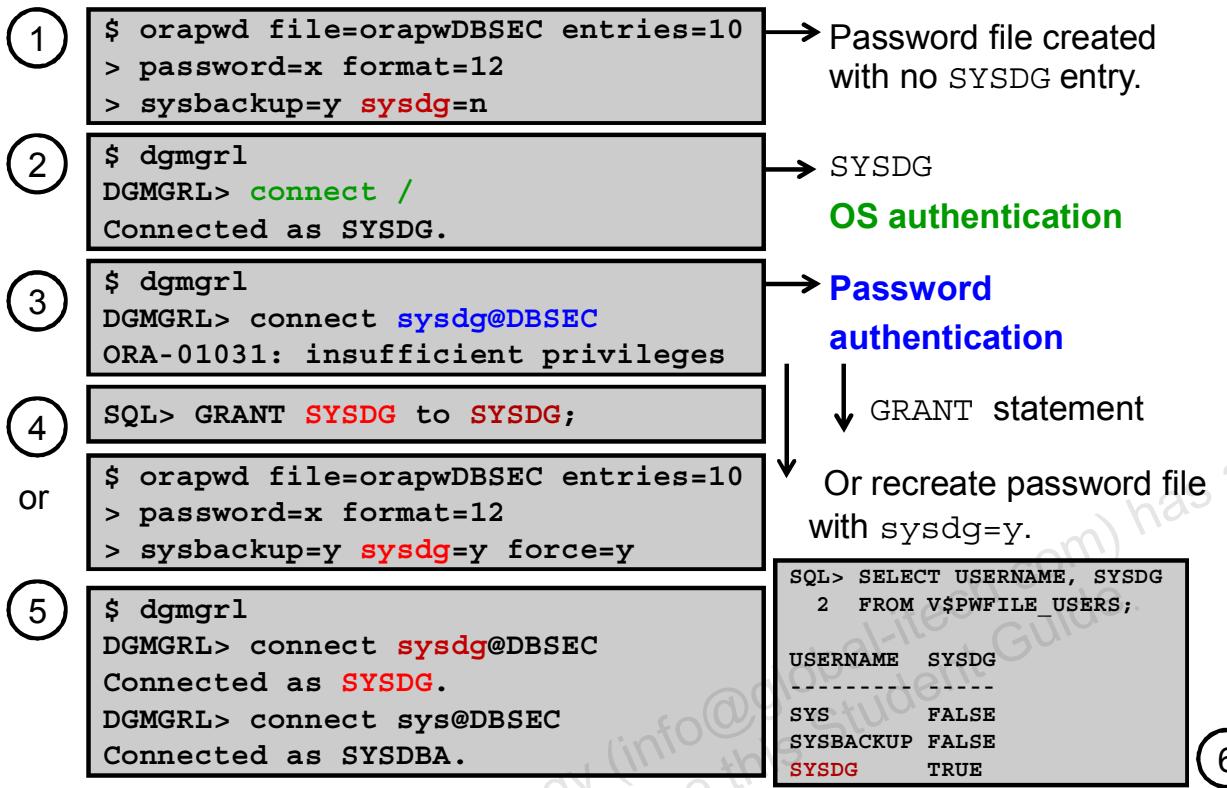
- **FORMAT:** If `12`, the default, the password file is created in a version12.x format and supports `SYSBACKUP`, `SYSDG`, and `SYSKM` administrative privileges. If `legacy`, the password file is created in legacy format, which is the format prior to Oracle Database 12c. You cannot set `FORMAT` to `legacy` when you specify the `SYSBACKUP` or the `SYSDG` argument.
- **SYSBACKUP:** If it is set to `y`, it creates a `SYSBACKUP` user entry in the password file, and you are prompted for the `SYSBACKUP` user password. The `SYSBACKUP` user password is stored in the created password file.
- **SYSDG:** If it is set to `y`, it creates a `SYSDG` entry in the password file, and you are prompted for the `SYSDG` password. The `SYSDG` password is stored in the created password file.
- **INPUT_FILE:** Name of the input password file. ORAPWD migrates the entries in the input file to a new password file. This argument can convert a password file from `legacy` format to `12` format.

It is recommended not to use the SYSBACKUP user to connect as SYSBACKUP. The DBAs that are responsible for backup/recovery would all share the same password. This is undesirable first in terms of accountability. Also, it would be impossible to take away this privilege from particular DBAs later (unless changing the password). It is better to create a database account designated for each DBA and grant the SYSBACKUP privilege if necessary. In this way, no password is shared, and SYSBACKUP can be revoked from any DBA without affecting other DBAs.

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/passwor@pdb1 AS SYSBACKUP"'
```

Password Authentication for SYSDG



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

Here is an example where the password file is created in 12 format and therefore supports SYSBACKUP and SYSDG administrative privileges in the password file. But SYSDG is set to n and so the SYSDG entry is not in the password file.

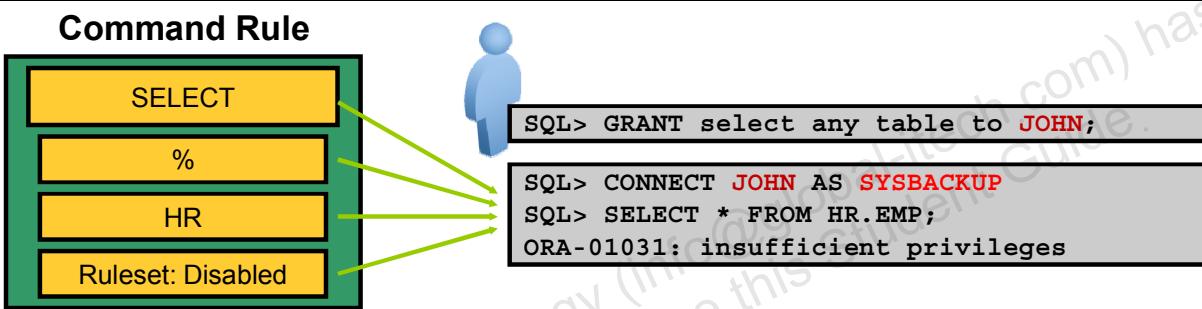
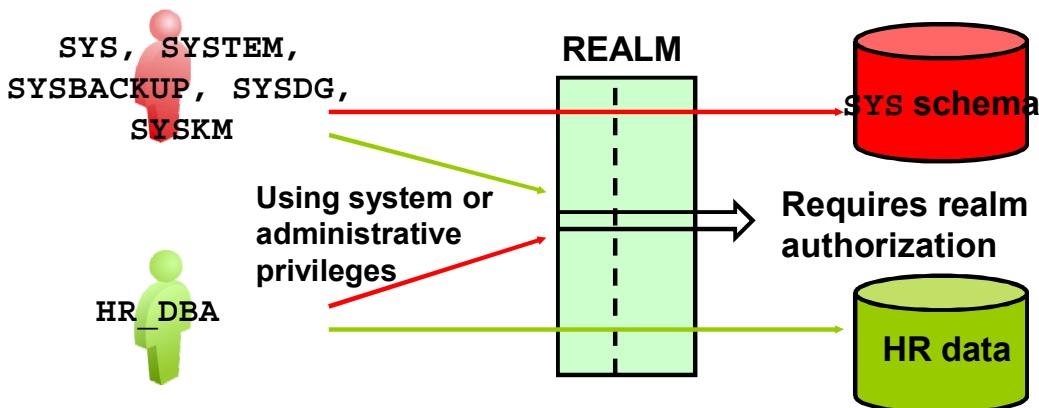
The connection with the DGMGRL utility and OS authentication is successful because the OS user is a member of the OSDG group.

Unlike SQL*Plus, where the user specifies "AS SYSxxx" at login time, the DGMGRL command-line interface accepts only a username/password pair. To seamlessly support the SYSDG administrative privilege, the command-line interface first attempts to log in by using SYSDG. If that login attempt fails, it tries SYSDBA.

The connection with password authentication fails because the entry for the SYSDG user is not in the password file. You can create the entry in the password file by the following two methods:

- Grant the SYSDG privilege to the SYSDG user.
- Re-create the password file in 12 format with the SYSDG argument specified as y.

Oracle Database Vault Data Protection and Administration Privileged Users



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

SYSBACKUP, SYSDG, SYSKM Privileges and Oracle Database Vault Protection

Oracle Database Vault provides two major mechanisms for enforcing separation of duty:

- Realms are a functional grouping of schemas, schema objects, or roles. They prevent users with system privileges from accessing the protected objects. For instance, users granted the SELECT ANY TABLE system privilege are not allowed to select a realm-secured table unless they are authorized for the realm.
- Command rules control the execution of a particular SQL command at run time. Specifically, a command that is protected by a command rule executes successfully only if the associated ruleset evaluates to TRUE.

Users with new administrative privileges are not exempt from the enforcement of realms or command rules.

To authorize users with the SYSBACKUP administrative privilege to access protected objects in a particular realm, the user needs to be authorized for the realm.

The example on the slide defines a command rule that prevents any SELECT on HR objects with the Disabled ruleset, including users with SYSBACKUP administrative privilege. This command rule applies even if the user was granted the SELECT ANY TABLE system privilege.

Privileged Administrators' Auditing



SYSDBA, SYSOPER , SYSBACKUP, SYSDG, SYSKM

In Unified Audit Database

All actions audited with or without audit policies enabled

- RMAN backup, restore, recover
- alter database
- select * from hr.employees
- create Database Vault realm
- expdp , impdp
- create restore point
- STARTUP, SHUTDOWN

Audit records generated

In Nonunified Audit Database

**AUDIT_SYS_OPERATIONS=true
AUDIT_FILE_DEST=directory**

CONNECT
STARTUP, SHUTDOWN,
ALTER DATABASE
+
SELECT, DML, DDL

**View
SYS.UNIFIED_AUDIT_TRAIL**

**OS audit directory
*.aud files**

ORACLE

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

The actions of the new administrative users are audited in the same way that the SYSDBA or SYSOPER users are audited, regardless of the type of audit used (unified audit or standard audit).

Quiz

Which user is connected when using the CONNECT TARGET / statement in Recovery Manager?

- a. SYSBACKUP
- b. SYS



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

Answer: b

AS SYSBACKUP clause is required if you want to be connected as SYSBACKUP user.

Quiz

The password file now has two formats to handle new administrative privileges.

- a. True
- b. False



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

Answer: a

New System Privilege: PURGE DBA_RECYLEBIN

Purging all recycled objects:

- Requires SYSDBA privilege

```
SQL> CONNECT / AS SYSDBA  
SQL> purge dba_recyclebin ← SQL Command
```

Pre-12c

- Does not require SYSDBA privilege
- Uses a new system privilege: PURGE DBA_RECYLEBIN

```
SQL> CONNECT system  
SQL> grant PURGE DBA_RECYLEBIN to dba_junior;
```

12c

```
SQL> CONNECT dba_junior  
SQL> purge dba_recyclebin;  
DBA Recyclebin purged.
```

12c

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

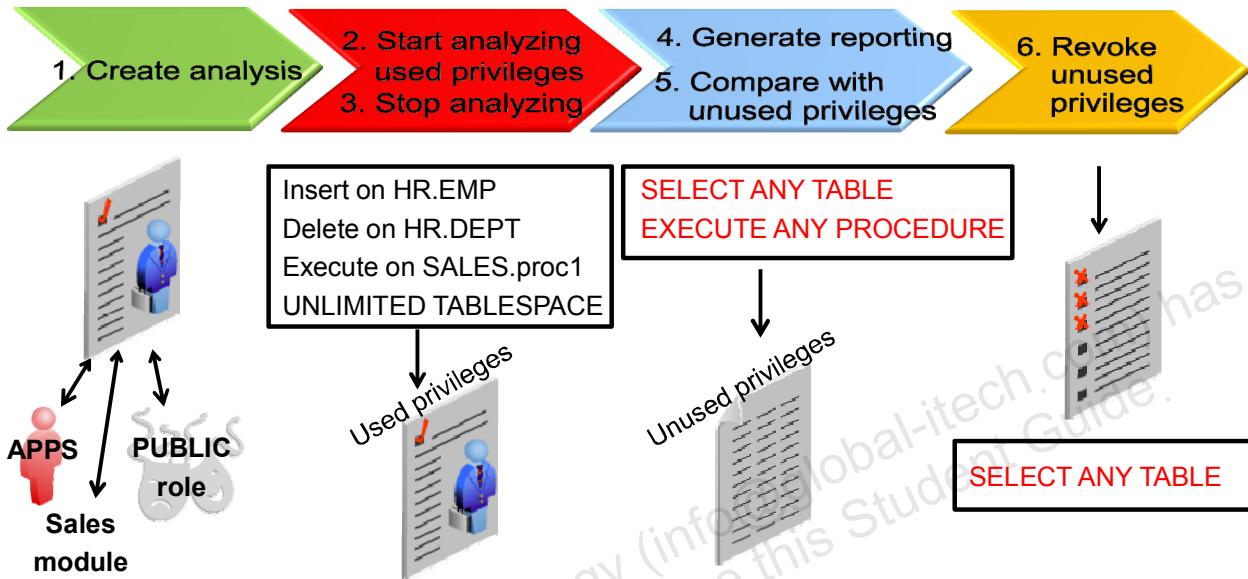
In Oracle Database 11g, the DBA uses the PURGE command to purge all objects. This operation requires the SYSDBA system privilege, which is highly undesirable in terms of separation of duty and least privilege.

In Oracle Database 12c, the same operation does not require the SYSDBA system privilege. It requires only the new PURGE DBA_RECYLEBIN system privilege.

Privilege Analysis

Increase database security: Revoke unused privileges

- Analyze used privileges to revoke unnecessary privileges.
- Use new package: DBMS_PRIVILEGE_CAPTURE



ORACLE

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

Excessive Privileges Challenge

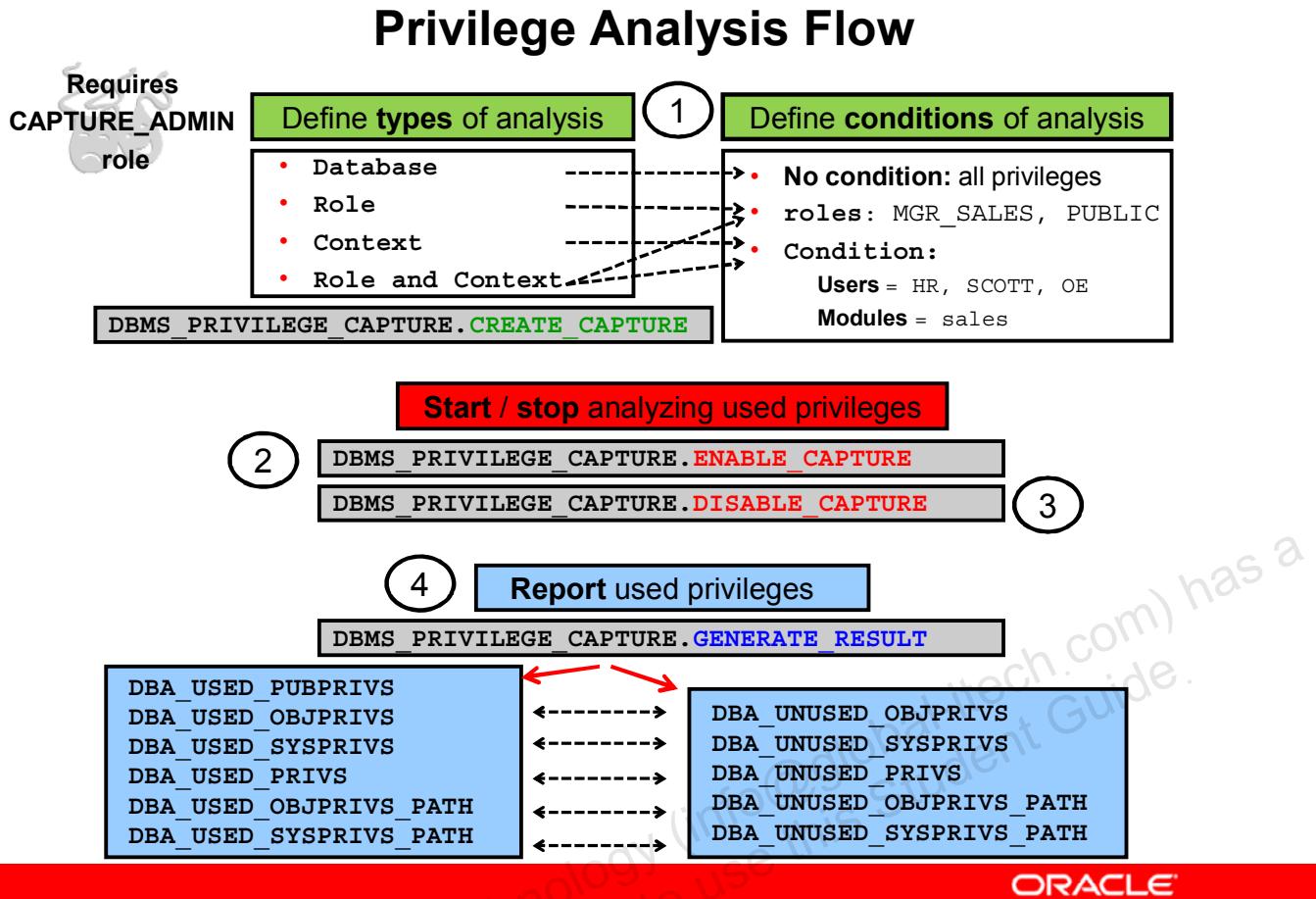
In addition to concern about the SYSDBA privilege usage, another major concern is that existing database and application users have excessive privileges. Excessive privileges violate the principle of least privilege.

To achieve the least privilege principle, unused privileges need to be identified.

Privilege Analysis

Oracle Database 12c offers a new package to analyze used privileges.

- You can use a privilege analysis policy to identify object and system privileges used to run an application module or to execute certain SQL statements or privileges used by defined roles.
- You can generate reports of used and unused privileges during the analysis period.
- The report helps the security officer revoke unnecessary privileges by comparing the used and unused granted privileges lists.



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

Privilege Analysis Types and Conditions

When creating an analysis, you first define the targeted objects to be analyzed in used privileges. You do that by setting the type of analysis:

- **Database analysis:** If no condition is given, it analyzes the used privileges (except privileges used by administrative users) within the whole database.
- **Role analysis:** If roles are defined, it analyzes the privileges exercised through any given role. For example, if you create a privilege analysis policy to analyze on PUBLIC, the privileges that are directly and indirectly granted to PUBLIC are analyzed when they are used.
- **Context-specific analysis:** If the contexts are defined, it analyzes the privileges that are used through a given application module or specified contexts.

Different conditions can be combined with “AND” and/or “OR” Boolean operators.

Because the created policy is not enabled by default, your next step is to enable the policy to start analyzing used privileges. After a certain time, you stop analyzing.

Reporting Used Privileges

Your third step is to generate a report. Reporting includes two types of results:

- Used privileges visible in DBA_USED_xxx and DBA_USED_xxx_PATH views
- Unused privileges visible in DBA_UNUSED_xxx and DBA_UNUSED_xxx_PATH views

Creating Policies: Database and Role Analysis

1. Create a policy to analyze used privileges.
 - Create a database analysis policy.

Create analysis

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2       name          => 'All_privs', -
  3       description   => 'Captures all privilege use', -
  4       type          => dbms_privilege_capture.g_database);
```

- Create a role analysis policy.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2       name          => 'Public_privs_capture', -
  3       description   => 'Privileges used by PUBLIC', -
  4       type          => dbms_privilege_capture.g_role, -
  5       roles         => role_name_list('PUBLIC'))
```

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2       name          => 'Audit_privs_capture', -
  3       description   => 'Privileges used by audit roles', -
  4       type          => dbms_privilege_capture.g_role, -
  5       roles         => role_name_list('AUDIT_ADMIN', 'AUDIT_VIEWER'))
```

ORACLE

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

The PL/SQL package used to create a privilege analysis policy is `SYS.DBMS_PRIVILEGE_CAPTURE`. To create a privilege analysis policy, use the procedure `CREATE_CAPTURE`.

- To create a database analysis policy, set the `TYPE` argument to the `dbms_privilege_capture.g_database` value.
- To create a role analysis policy, set the `TYPE` argument to `dbms_privilege_capture.g_role` and the `ROLES` argument to the list of the roles to be analyzed.

The second example on the slide shows how to set a policy to analyze all privileges that are used by `PUBLIC`.

The third example shows how to set a policy to analyze all privileges that are used through the `AUDIT_ADMIN` and `AUDIT_VIEWER` roles.

Creating Policies: Context Analysis

- Create a context analysis policy.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -  
 2      name    => 'Privils_HR_OE_logged_users', -  
 3      description => 'All privileges used by HR,OE', -  
 4      type    => dbms_privilege_capture.g_context, -  
 5      condition => -  
 6          'SYS_CONTEXT(''USERENV'', ''SESSION_USER'')=''HR'' -  
 7      OR -  
 8          SYS_CONTEXT(''USERENV'', ''SESSION_USER'')='''OE'''')
```

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -  
 2      name    => 'Privils_AcctPayable_capture', -  
 3      description => 'All privileges used by module', -  
 4      type    => dbms_privilege_capture.g_context, -  
 5      condition => 'SYS_CONTEXT -  
 6 (''USERENV'', ''MODULE'')='''Account Payable'''')
```



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

To create a context analysis policy, set the **TYPE** argument to the `dbms_privilege_capture.g_context` value. Also set a value of the analyzed contexts in the **CONDITION** argument.

The first example on the slide shows how to set a policy to analyze all privileges used by the HR and OE connected users. Notice that two conditions are combined with the **OR** operator.

The second example shows how to set a policy to analyze all privileges that are used when you run the Account Payable module.

Creating Policies: Combined Analysis Types

- Create a policy combining two analysis types.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE ( -
  2      name  => 'Privils_context_role', -
  3      description => 'Captures Context and role', -
  4      type => dbms_privilege_capture.g_role_and_context, -
  5      roles  => role_name_list('PUBLIC')
  6      condition => 'SYS_CONTEXT -
  7 (''USERENV'', ''MODULE'')='Account Payable''')
```



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

To create a role and context analysis type policy, set the `TYPE` argument to the `dbms_privilege_capture.g_role_and_context` value. Also define values for the `ROLES` argument and `CONDITION` for the context values.

The example on the slide shows how to set a policy to analyze all privileges that are used by PUBLIC when you run the Account Payable module.

Analyzing and Reporting

2. Start the analysis of used privileges.



```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ( -
2      name      => 'All_privs')
```

3. After some time, stop analyzing.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ( -
2      name      => 'All_privs')
```

4. Generate the report.

```
SQL> exec SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ( -
2      name      => 'All_privs')
```



ORACLE

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

Capturing Used Privileges

To start the analysis, use the `ENABLE_CAPTURE` procedure.

Reporting

To generate the results of the started analysis, first stop the analysis by using the `DISABLE_CAPTURE` procedure, and then use the `GENERATE_RESULT` procedure.



Compare with
granted unused
privileges

SYSTEM and OBJECT Used Privileges

- View SYSTEM privileges used during the entire analysis.

```
SQL> select USERNAME, SYS_PRIV from DBA_USED_SYSPRIVS;

USERNAME      SYS_PRIV
-----
TOM           CREATE SESSION
OE            UPDATE ANY TABLE
OE            CREATE SESSION
JIM           CREATE SESSION
```

- View OBJECT privileges used during the entire analysis.

```
SQL> select USERNAME, OBJECT_OWNER, OBJECT_NAME, OBJ_PRIV
  2  from DBA_USED_OBJPRIVS where username in ('JIM','TOM');

USERNAME      OBJECT_OWNER OBJECT_NAME          OBJ_PRIV
-----
JIM           SYS          DBMS_APPLICATION_INFO EXECUTE
JIM           HR           EMPLOYEES             DELETE
TOM           SH           SALES                 SELECT
```



ORACLE

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

Reporting by Using Dictionary Views

When you generate the analysis results, SYSTEM and OBJECT used privileges populate the following views:

- DBA_USED_SYSPRIVS : The first example on the slide shows that TOM , JIM, and OE connected to the database by using the CREATE SESSION system privilege, and OE updated a table by using the UPDATE ANY TABLE system privilege.
- DBA_USED_OBJPRIVS : The second example shows that JIM executed the SYS.DBMS_APPLICATION_INFO procedure by using the EXECUTE object privilege, and he deleted rows from HR.EMPLOYEES table by using the DELETE object privilege. TOM selected rows from SH.SALES table by using the SELECT object privilege.

Used Privileges Results

- View SYSTEM and OBJECT used privileges:

SQL> select USERNAME, SYS_PRIV, OBJ_PRIV, OBJECT_OWNER, OBJECT_NAME 2 from DBA_USED_PRIVS ;				
USERNAME	SYS_PRIV	OBJ_PRIV	OBJECT_OWNER	OBJECT_NAME
JIM		SELECT	HR	EMPLOYEES
JIM		DELETE	HR	EMPLOYEES
TOM		CREATE SESSION		
TOM		SELECT	SH	SALES
OE		UPDATE	HR	DEPARTMENTS

- View the path for OBJECT used privileges:

SQL> select USERNAME, OBJ_PRIV, OBJECT_NAME, PATH 2 from DBA_USED_OBJPRIVS_PATH where username in ('TOM', 'JIM')			
USERNAME	OBJ_PRIV	OBJECT_NAME	PATH
OE	UPDATE	DEPARTMENTS	GRANT_PATH('OE')
JIM	DELETE	EMPLOYEES	GRANT_PATH('JIM', 'HR_MGR')
JIM	SELECT	EMPLOYEES	GRANT_PATH('JIM', 'HR_MGR')
TOM	SELECT	SALES	GRANT_PATH('TOM', 'SALES_CLERK')

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

Reporting

You can view all SYSTEM and OBJECT used privileges in the DBA_USED_PRIVS view.

If you need to know how the privileges were granted to the users, display the PATH column from DBA_USED_OBJPRIVS_PATH and DBA_USED_SYSPRIVS_PATH.

The second example reveals the following information:

- OE updated rows from the DEPARTMENTS table because he is directly granted the OBJECT privilege UPDATE.
- JIM deleted rows from the EMPLOYEES table because he is granted the OBJECT privilege DELETE through the HR_MGR role.
- JIM selected rows from the EMPLOYEES table because he is granted the OBJECT privilege SELECT through the HR_MGR role.
- TOM selected rows from the SALES table because he is granted the OBJECT privilege SELECT through the SALES_CLERK role.

Compare Used and Unused Privileges

Compare with
granted unused
privileges

- View SYSTEM and OBJECT **used** privileges:

```
SQL> select USERNAME, SYS_PRIV, OBJ_PRIV, OBJECT_OWNER, OBJECT_NAME
  2  from DBA_USED_PRIVS where username='JIM';

USERNAME  SYS_PRIV  OBJ_PRIV   OBJECT_OWNER  OBJECT_NAME
-----  -----  -----
JIM          SELECT      HR        EMPLOYEES
JIM          DELETE      HR        EMPLOYEES
```

- View SYSTEM and OBJECT **unused** privileges:

```
SQL> select USERNAME, OBJ_PRIV, OBJECT_NAME, PATH
  2  from DBA_UNUSED_PRIVS where username='JIM';

USERNAME  OBJ_PRIV  OBJECT_NAME  PATH
-----  -----  -----
JIM        INSERT    EMPLOYEES  GRANT_PATH('JIM','HR_MGR')
JIM        UPDATE    EMPLOYEES  GRANT_PATH('JIM','HR_MGR')
```

- Decide if unused privileges need to be revoked

Revoke
unused
privileges

ORACLE

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

Comparing to Revoke Unnecessary Privileges

You can view all unused SYSTEM and OBJECT privileges granted in the DBA_UNUSED_PRIVS view.

If you compare the list of used and unused privileges, you can identify the privileges that are granted but are not used, and you can decide whether to revoke the unused privileges.

For the example on the slide, JIM used the SELECT and DELETE privileges on the HR.EMPLOYEES table, and he did not use INSERT or UPDATE on the same table. The INSERT and UPDATE privileges are granted through the HR_MGR role.

Views

- List of analyses:

NAME	TYPE	ENA	ROLES
<hr/>			
CONTEXT			
→ All_privs	DATABASE	N	
→ Public_privs	ROLE	N	ROLE_ID_LIST(1)
→ HR_SH_privs	ROLE	Y	ROLE_ID_LIST(112, 113)
→ Privils_HR_OE_logged CONTEXT		N	
	SYS_CONTEXT('USERENV', 'SESSION_USER') = 'HR' OR		
	SYS_CONTEXT('USERENV', 'SESSION_USER') = 'OE'		
→ HR_Sales_role	ROLE_AND_CONTEXT	N	ROLE_ID_LIST(113)
	SYS_CONTEXT('USERENV', 'SESSION_USER') = 'HR'		

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

Dictionary View

To view the list of created analyses, use the DBA_PRIV_CAPTURES dictionary view. The TYPE column can hold four values: DATABASE, ROLE, CONTEXT, and ROLE_AND_CONTEXT. The ENABLED column is set to Y when the analysis is analyzing used privileges. The ROLES column holds the list of roles, and the CONTEXT column holds the condition. The roles and the condition are defined in the creation of the analysis policy.

If the database is a CDB, you create, start, stop, and generate reports in the that you are connected to.

This means that an analysis collects information from the sessions of either the root or a PDB where you created and started the analysis policy. It does not collect information for all containers during an analysis.

Dropping an Analysis

1. Disable the analysis.

```
SQL> exec dbms_privilege_capture.DROP_CAPTURE('Capture1')
BEGIN dbms_privilege_capture.DROP_CAPTURE('Capture1'); END;

*
ERROR at line 1:
ORA-47932: Privilege capture Capture1 is still enabled.
ORA-06512: at "SYS.DBMS_PRIVILEGE_CAPTURE", line 82
ORA-06512: at line 1

SQL> exec dbms_privilege_capture.DISABLE_CAPTURE('Capture1')
PL/SQL procedure successfully completed.
```

2. Drop the analysis.

```
SQL> exec dbms_privilege_capture.DROP_CAPTURE('Capture1')
PL/SQL procedure successfully completed.
```



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

To drop an analysis, first disable the policy if it was already started. Dropping a policy also drops all used and unused privilege records that are associated with this privilege policy.

Quiz

To revoke unnecessary and unused privileges granted, use the Privilege Analysis. Is the sequence in the proper order? True or False?

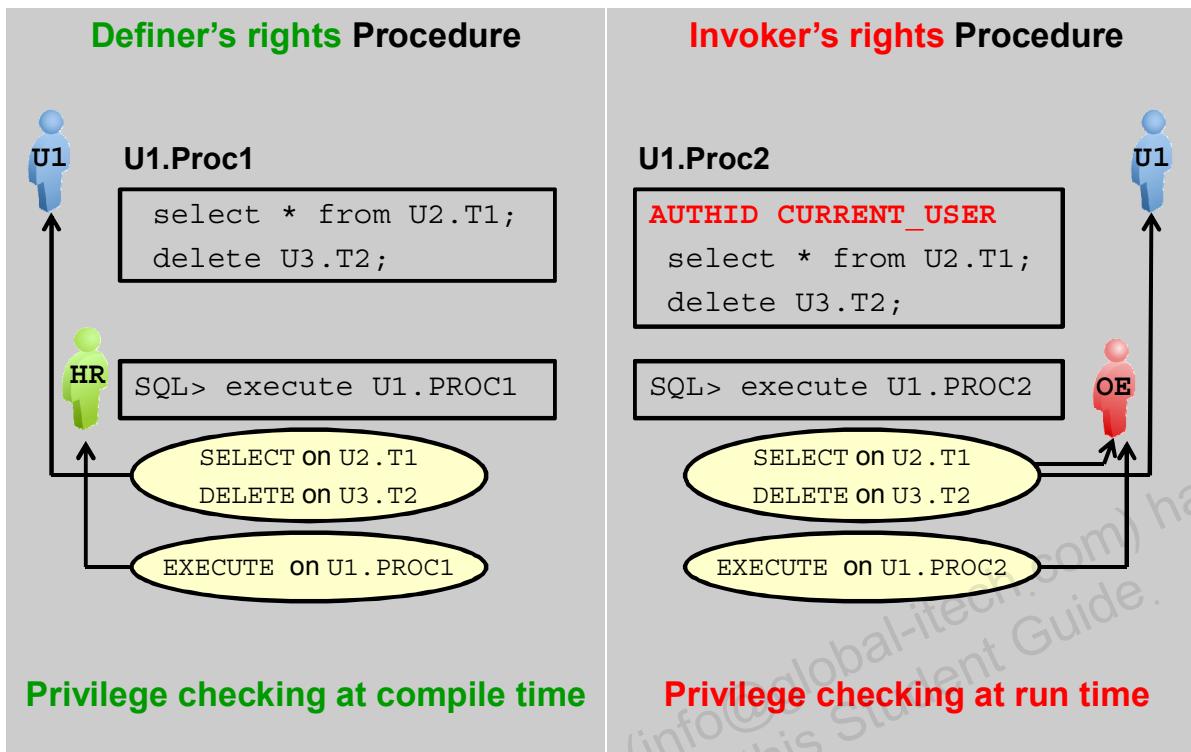
- a. Set up the analysis policy type (database, role, context).
- b. Start the analysis.
- c. Stop the analysis.
- d. Generate the results.
- e. View the results in DBA_USED_PRIVS and DBA_UNUSED_PRIVS.



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

Answer: True

Privilege Checking During PL/SQL Calls



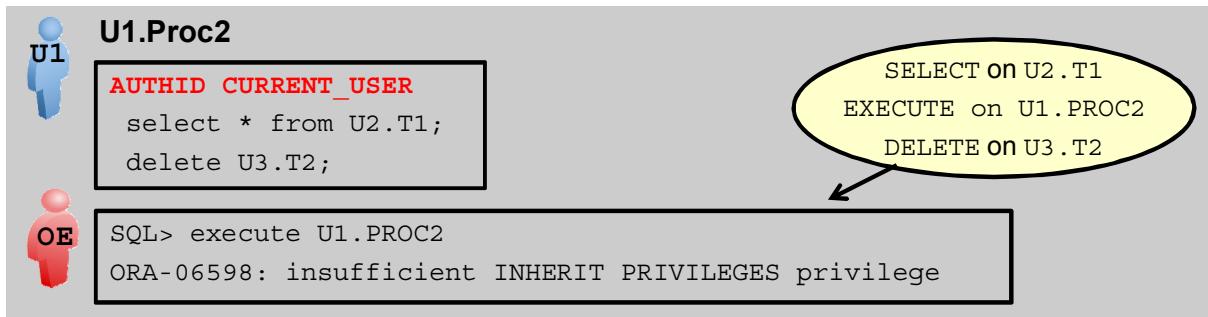
ORACLE

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

In Oracle Database 11g, the AUTHID property of a stored PL/SQL unit affects the name resolution and privilege checking of SQL statements that the unit issues at run time.

- A unit whose AUTHID value is DEFINER (the default) is called a definer's rights unit. The invocation of **U1.PROC1** by **HR** succeeds because the procedure runs with the privileges of its owner (**U1**).
- A unit whose AUTHID value is CURRENT_USER is called an invoker's rights unit. The invocation of **U1.PROC2** by **OE** succeeds because the procedure runs with the privileges of the current user (**OE**), not the owner (**U1**). In this case, the privilege checking is performed at run time.

New Privilege Checking During PL/SQL Calls



- Additional required privilege checking at run time:
 - Of a database user passing into an AUTHID CURRENT_USER PL/SQL routine
 - Of a database user passing through an AUTHID CURRENT_USER “callspec” over a C or Java routine
- INHERIT PRIVILEGES object privilege
- INHERIT ANY PRIVILEGES system privilege

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

Risks

A low privilege user could own an invoker's rights procedure that could potentially perform unintended or malicious actions if it is executed by a high-privileged user.

An invoker's right procedure can perform inappropriate actions if it is invoked by another procedure that does not expect an invoker's rights procedure.

In Oracle Database 11g, the caller to a procedure had no control over who accessed the caller's privileges. Only the owner of the procedure controlled the right's inheritance.

New Privilege Checking

Privilege checking in Oracle Database 12c implements a new restriction, not a new power. Existing cases that did not require a privilege check now require one. When a user runs an invoker's rights procedure, Oracle Database checks the procedure owner's privileges before initiating or running the code. The owner must have the INHERIT PRIVILEGES object privilege on the invoking user or the INHERIT ANY PRIVILEGES privilege. If this is not the case, the runtime system raises an error.

The session is temporarily switched into an environment that treats the entered routine as the definer's rights. It then checks that it has the INHERIT PRIVILEGES object privilege on the caller's active current user or that it has the INHERIT ANY PRIVILEGES system privilege. The session then reverts to its prior environment. This treatment of the routine as definer's rights mirrors the treatment of the routine during compilation.

INHERIT (ANY) PRIVILEGES Privileges

- Invoking users can control who can access their privileges when they run an invoker's rights procedure.
- Invoking users grant the INHERIT PRIVILEGES object privilege only to **trusted users** (object is a **USER**).

```
SQL> CONNECT oe
SQL> grant INHERIT PRIVILEGES ON USER oe TO u1;
```

```
SQL> select PRIVILEGE, TYPE, TABLE_NAME, GRANTEE
  2  from DBA_TAB_PRIVS where grantee='U1';
```

PRIVILEGE	GRANTEE	TYPE	TABLE_NAME	GRANTEE
INHERIT PRIVILEGES	USER	OE		U1

- Newly created users are granted INHERIT PRIVILEGES object privilege on themselves through PUBLIC.
- INHERIT ANY PRIVILEGES means on all users.

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

INHERIT PRIVILEGES Object Privilege

The benefit of this privilege is that it gives invoking users control over who can access their privileges when they run an invokers' rights procedure.

```
SQL> CONNECT invoking_user
SQL> GRANT INHERIT PRIVILEGES ON USER inv_user TO proc_owner;
```

By default, when a CREATE USER creates a new database-defined user, INHERIT PRIVILEGES privilege on that user is made PUBLIC, with the user itself listed as the grantor.

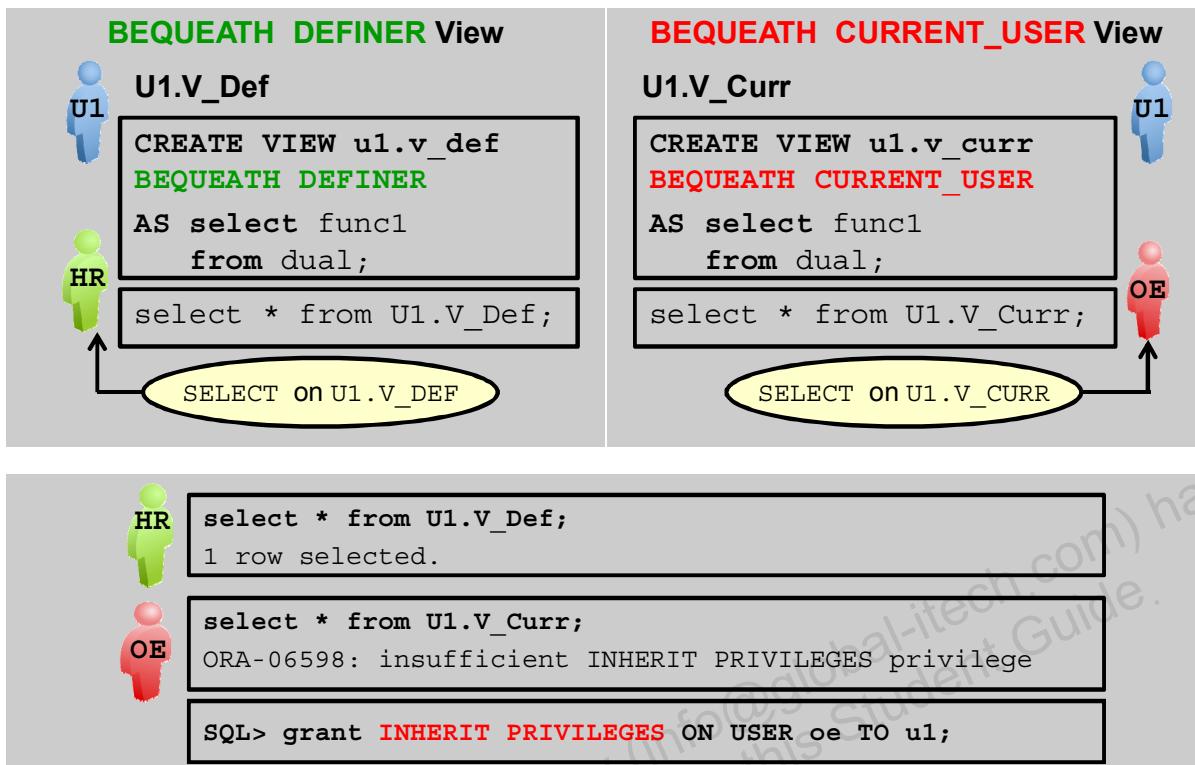
```
SQL> select PRIVILEGE, TYPE, TABLE_NAME, GRANTEE
  2  from DBA_TAB_PRIVS where type='USER' and table_name='X';

PRIVILEGE          TYPE      TABLE_NAME      GRANTEE
-----          -----      -----      -----
INHERIT PRIVILEGES    USER           X        PUBLIC
```

INHERIT ANY PRIVILEGES System Privilege

A user being granted the INHERIT ANY PRIVILEGES system privilege inherits privileges on all users.

Privilege Checking with New BEQUEATH Views



ORACLE

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

BEQUEATH Views

Oracle Database 12c introduces `BEQUEATH CURRENT_USER` views, which bring a security benefit to objects of this type similar to stored PL/SQL units. These views partially behave like invoker's rights rather than owner's rights. That is, when you call an `AUTHID CURRENT_USER` function or an invoker's rights PL/SQL or Java function, the current schema, current user, and currently enabled roles within the operation's execution can be inherited from the querying user's environment. `BEQUEATH CURRENT_USER` views are only a subset of the behavior of invoker's rights in this release.

The `BEQUEATH CURRENT_USER` views are in contrast to the `BEQUEATH DEFINER` behavior of existing views.

The `BEQUEATH` view type is displayed in a new `BEQUEATH` column in `DBA_VIEWS`.

INHERIT PRIVILEGES and BEQUEATH CURRENT_USER Views

The owner of a `BEQUEATH CURRENT_USER` view must have the `INHERIT PRIVILEGES` object privilege on the invoking user or the `INHERIT ANY PRIVILEGES` system privilege.

Quiz

Select the statement that is true when you execute an invoker's rights procedure.

- a. The calling user must be granted the INHERIT PRIVILEGES object privilege on the user owner of the procedure.
- b. The calling user must grant the INHERIT PRIVILEGES object privilege on the user owner of the procedure.
- c. The owner of the procedure must grant the INHERIT PRIVILEGES object privilege on the calling user.
- d. The owner of the procedure must be granted the INHERIT PRIVILEGES object privilege on the calling user.



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

Answer: d

Quiz

When you select from a BEQUEATH CURRENT_USER view, select the statement that is true when you call an AUTHID CURRENT_USER function.

- a. The current schema, user, and enabled roles are inherited from the querying user's environment.
- b. The current schema, user, and enabled roles are set to the owner of the view.



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

Answer: a

Summary

In this lesson, you should have learned how to:

- Use new administrative privileges like SYSBACKUP
- Create the password file to support new privileges
- Use new system privilege PURGE DBA_RECYLEBIN
- Use database privilege analysis
- Create and enable privilege analysis
- View capture results to decide to revoke unused privileges
- Control privileges for invoker's rights procedures
- Use INHERIT PRIVILEGES privilege
- Create new BEQUEATH views
- Manage the invoker's rights behavior for BEQUEATH views



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

Practice 10 Overview: Privileges

These practices cover the following topics:

- Using new SYSBACKUP administrative privilege
- Using Privilege Analysis
 - For all users
 - For roles (*optional*)
 - For specific role and context (*optional*)
- Using INHERIT PRIVILEGES object privilege on AUTHID CURRENT_USER procedures (*optional*)
- Using BEQUEATH CURRENT_USER views (*optional*)



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

