

Hardware and Software
Engineered to Work Together

Oracle Database 12c: Administration Workshop

Student Guide – Volume I
D78846GC20
Edition 2.0 | December 2014 | D89298

Learn more from Oracle University at oracle.com/education/

Authors

Donna K. Keesling
James L. Spiller

Technical Contributors and Reviewers

Daryl Balaski
Rainer Bien
Maria Billings
Andy Fortunak
Joel Goodman
Daniela Hansell
Pat Huey
Dominique Jeunot
Gwen Lazenby
Ira Singer
Lori Tritz
Branislav Valny
Harald Van Breederode

Editors

Vijayalakshmi Narasimhan
Malavika Jinka

Publishers

Veena Narasimhan
Jayanthi Keshavamurthy

Copyright © 2014, 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

1 Introduction

- Objectives 1-2
- Course Objectives 1-3
- Suggested Schedule 1-4
- Oracle Database Innovation 1-5
- Enterprise Cloud Computing 1-6
- Course Examples: HR Sample Schema 1-7
- Summary 1-8

2 Exploring Oracle Database Architecture

- Objectives 2-2
- Oracle Database Server Architecture: Overview 2-3
- Oracle Database Instance Configurations 2-4
- Connecting to the Database Instance 2-5
- Oracle Database Memory Structures 2-6
- Shared Pool 2-8
- Database Buffer Cache 2-10
- Redo Log Buffer 2-11
- Large Pool 2-12
- Java Pool 2-13
- Streams Pool 2-14
- Program Global Area (PGA) 2-15
- In-Memory Column Store: Introduction 2-16
- In-Memory Column Store: Overview 2-18
- Full Database In-Memory Caching 2-20
- Quiz 2-22
- Process Architecture 2-24
- Process Structures 2-26
- Database Writer Process (DBWn) 2-28
- Log Writer Process (LGWR) 2-30
- Checkpoint Process (CKPT) 2-32
- System Monitor Process (SMON) 2-33
- Process Monitor Process (PMON) 2-34
- Recoverer Process (RECO) 2-35
- Listener Registration Process (LREG) 2-36

Archiver Processes (ARCn)	2-37
Database Storage Architecture	2-38
Logical and Physical Database Structures	2-40
Segments, Extents, and Blocks	2-42
Tablespaces and Data Files	2-43
SYSTEM and SYSAUX Tablespaces	2-44
Oracle Container Database: Introduction	2-45
Multitenant Architecture	2-46
Automatic Storage Management	2-47
ASM Storage Components	2-48
Interacting with an Oracle Database: Memory, Processes, and Storage	2-49
Quiz	2-51
Summary	2-52
Practice: Overview	2-53

3 Oracle Database Management Tools

Objectives	3-2
Oracle Database Management Tools: Introduction	3-3
Using SQL*Plus	3-4
Calling SQL*Plus from a Shell Script	3-5
Calling a SQL Script from SQL*Plus	3-6
Oracle SQL Developer: Connections	3-7
Oracle SQL Developer: DBA Actions	3-8
Oracle Enterprise Manager Database Express Architecture	3-9
Configuring Enterprise Manager Database Express	3-10
Logging In to Oracle Enterprise Manager Database Express	3-11
Using the Database Home Page	3-12
Using Enterprise Manager Database Express Menus	3-13
Oracle Enterprise Manager Cloud Control Components	3-14
Controlling the Enterprise Manager Cloud Control Framework	3-15
Starting the Enterprise Manager Cloud Control Framework	3-16
Stopping the Enterprise Manager Cloud Control Framework	3-17
Types of Enterprise Manager Cloud Control Targets	3-18
Enterprise Manager Cloud Control	3-19
Using Enterprise Manager Cloud Control	3-20
Quiz	3-21
Summary	3-22
Practice: Overview	3-23

4 Managing the Database Instance
Objectives 4-2
Initialization Parameter Files 4-3
Types of Initialization Parameters 4-5
Initialization Parameters: Examples 4-6
Using SQL*Plus to View Parameters 4-10
Changing Initialization Parameter Values 4-13
Changing Parameter Values: Examples 4-15
Quiz 4-16
Starting Up an Oracle Database Instance: NOMOUNT 4-17
Starting Up an Oracle Database Instance: MOUNT 4-18
Starting Up an Oracle Database Instance: OPEN 4-19
Startup Options: Examples 4-20
Shutdown Modes 4-21
Shutdown Options 4-22
Shutdown Options: Examples 4-25
Viewing the Alert Log 4-26
Using Trace Files 4-28
Administering the DDL Log File 4-30
Understanding the Debug Log File 4-31
Using Dynamic Performance Views 4-32
Dynamic Performance Views: Usage Examples 4-33
Dynamic Performance Views: Considerations 4-34
Data Dictionary: Overview 4-35
Data Dictionary Views 4-36
Data Dictionary: Usage Examples 4-39
Quiz 4-40
Summary 4-41
Practice: Overview 4-42

5 Configuring the Oracle Network Environment

Objectives 5-2
Oracle Net Services: Overview 5-3
Oracle Net Listener: Overview 5-4
Establishing Oracle Network Connections 5-5
Connecting to an Oracle Database 5-6
Name Resolution 5-7
Establishing a Connection 5-8
User Sessions 5-9
Naming Methods 5-11
Easy Connect 5-12

Local Naming 5-13
Directory Naming 5-14
External Naming Method 5-15
Tools for Configuring and Managing Oracle Net Services 5-16
Defining Oracle Net Services Components 5-17
Using Enterprise Manager Cloud Control 5-18
Using Oracle Net Manager 5-19
Using Oracle Net Configuration Assistant 5-20
Using the Listener Control Utility 5-21
Listener Control Utility Syntax 5-22
Advanced Connection Options 5-24
Testing Oracle Net Connectivity 5-26
Comparing Dedicated Server and Shared Server Configurations 5-27
User Sessions: Dedicated Server Process 5-28
User Sessions: Shared Server Processes 5-29
SGA and PGA Usage 5-30
Shared Server Configuration Considerations 5-31
Configuring Communication Between Databases 5-32
Connecting to Another Database 5-33
Quiz 5-34
Summary 5-36
Practice: Overview 5-37

6 Administering User Security

Objectives 6-2
Database User Accounts 6-3
Predefined Administrative Accounts 6-5
Administrative Privileges 6-6
Protecting Privileged Accounts 6-7
Authenticating Users 6-8
Administrator Authentication 6-10
OS Authentication and OS Groups 6-11
Managing Users 6-13
Creating a User 6-14
Unlocking a User Account and Resetting the Password 6-15
Privileges 6-16
System Privileges 6-17
Revoking System Privileges with ADMIN OPTION 6-19
Granting Object Privileges 6-20
Object Privileges 6-21
Revoking Object Privileges with GRANT OPTION 6-22

Using Roles to Manage Privileges	6-23
Assigning Privileges to Roles and Assigning Roles to Users	6-24
Predefined Roles	6-25
Creating a Role	6-26
Secure Roles	6-27
Assigning Roles to Users	6-28
Privilege Analysis	6-29
Privilege Analysis Flow	6-30
Profiles and Users	6-31
Implementing Password Security Features	6-33
Creating a Password Profile	6-35
Supplied Password Verification Functions	6-36
Modifications to the Default Profile	6-37
Assigning Quotas to Users	6-38
Applying the Principle of Least Privilege	6-40
Quiz	6-42
Summary	6-46
Practice: Overview	6-47

7 Managing Database Storage Structures

Objectives	7-2
How Table Data Is Stored	7-3
Database Block: Contents	7-4
Exploring the Storage Structure	7-5
Creating a New Tablespace	7-6
Tablespaces Created by Default: Overview	7-12
Altering a Tablespace	7-14
Adding a Data File to a Tablespace	7-16
Making Changes to a Data File	7-17
Dropping Tablespaces	7-18
Viewing Tablespace Information	7-19
Oracle Managed Files (OMF)	7-20
Quiz	7-22
Enlarging the Database	7-23
Moving or Renaming an Online Data File	7-24
Summary	7-26
Practice: Overview	7-27

8 Managing Space

- Objectives 8-2
- Space Management: Overview 8-3
- Block Space Management 8-4
- Row Chaining and Migration 8-5
- Quiz 8-7
- Free Space Management Within Segments 8-8
- Types of Segments 8-9
- Allocating Extents 8-10
- Understanding Deferred Segment Creation 8-11
- Viewing Deferred Segment Information 8-12
- Controlling Deferred Segment Creation 8-13
- Restrictions and Exceptions 8-14
- Additional Automatic Functionality 8-15
- Quiz 8-16
- Table Compression: Overview 8-17
- Compression for Direct-Path Insert Operations 8-18
- Advanced Row Compression for DML Operations 8-20
- Specifying Table Compression 8-21
- Using the Compression Advisor 8-22
- Using the DBMS_COMPRESSION Package 8-23
- Proactive Tablespace Monitoring 8-24
- Thresholds and Resolving Space Problems 8-25
- Monitoring Tablespace Space Usage 8-26
- Shrinking Segments 8-27
- Results of Shrink Operation 8-28
- Reclaiming Space Within ASSM Segments 8-29
- Using the Segment Advisor 8-30
- Automatic Segment Advisor 8-31
- Shrinking Segments by Using SQL 8-32
- Shrinking Segments by Using Enterprise Manager 8-33
- Managing Resumable Space Allocation 8-34
- Using Resumable Space Allocation 8-35
- Resuming Suspended Statements 8-37
- What Operations Are Resumable? 8-39
- Quiz 8-40
- Summary 8-41
- Practice: Overview 8-42

9 Managing Undo Data

- Objectives 9-2
- Undo Data: Overview 9-3
- Transactions and Undo Data 9-5
- Storing Undo Information 9-6
- Comparing Undo Data and Redo Data 9-7
- Managing Undo 9-8
- Configuring Undo Retention 9-9
- Categories of Undo 9-10
- Guaranteeing Undo Retention 9-11
- Changing an Undo Tablespace to a Fixed Size 9-12
- Temporary Undo: Overview 9-13
- Temporary Undo: Benefits 9-14
- Enabling Temporary Undo 9-15
- Monitoring Temporary Undo 9-16
- Viewing Undo Information 9-17
- Viewing Undo Activity 9-18
- Using the Undo Advisor 9-19
- Quiz 9-20
- Summary 9-21
- Practice: Overview 9-22

10 Managing Data Concurrency

- Objectives 10-2
- Locks 10-3
- Locking Mechanism 10-4
- Data Concurrency 10-5
- DML Locks 10-7
- Enqueue Mechanism 10-8
- Lock Conflicts 10-9
- Possible Causes of Lock Conflicts 10-10
- Detecting Lock Conflicts 10-11
- Resolving Lock Conflicts 10-12
- Resolving Lock Conflicts by Using SQL 10-13
- Deadlocks 10-14
- Quiz 10-15
- Summary 10-17
- Practice: Overview 10-18

11 Implementing Oracle Database Auditing

- Objectives 11-2
- Database Security 11-3
- Monitoring for Compliance 11-5
- Types of Activities to be Audited 11-6
- Mandatorily Audited Activities 11-7
- Understanding Auditing Implementation 11-8
- Administering the Roles Required for Auditing 11-9
- Database Auditing: Overview 11-10
- Understanding the Audit Architecture 11-11
- Enabling Unified Auditing 11-12
- Configuring Auditing 11-13
- Using Enterprise Manager Cloud Control 11-14
- Creating a Unified Audit Policy 11-15
- Creating an Audit Policy: System-Wide Audit Options 11-16
- Creating an Audit Policy: Object-Specific Actions 11-17
- Creating an Audit Policy: Specifying Conditions 11-18
- Enabling and Disabling Audit Policies 11-19
- Altering a Unified Audit Policy 11-20
- Viewing Audit Policy Information 11-21
- Setting the Write Mode for Audit Trail Records 11-22
- Value-Based Auditing 11-23
- Fine-Grained Auditing 11-25
- FGA Policy 11-26
- Audited DML Statement: Considerations 11-28
- FGA Guidelines 11-29
- Archiving and Purging the Audit Trail 11-30
- Purging Audit Trail Records 11-31
- Quiz 11-32
- Summary 11-33
- Practice: Overview 11-34

12 Backup and Recovery: Concepts

- Objectives 12-2
- DBA Responsibilities 12-3
- Categories of Failure 12-5
- Statement Failure 12-6
- User Process Failure 12-7
- Network Failure 12-8
- User Error 12-9
- Flashback Technology 12-10

Instance Failure	12-12
Understanding Instance Recovery: Checkpoint (CKPT) Process	12-13
Understanding Instance Recovery: Redo Log Files and Log Writer	12-14
Understanding Instance Recovery	12-15
Phases of Instance Recovery	12-16
Tuning Instance Recovery	12-17
Using the MTTR Advisor	12-18
Media Failure	12-19
Comparing Complete and Incomplete Recovery	12-20
Complete Recovery Process	12-21
Point-in-Time Recovery Process	12-22
Oracle Data Protection Solutions	12-24
Quiz	12-25
Summary	12-26

13 Backup and Recovery: Configuration

Objectives	13-2
Configuring for Recoverability	13-3
Configuring the Fast Recovery Area	13-4
Monitoring the Fast Recovery Area	13-5
Multiplexing Control Files	13-6
Redo Log Files	13-8
Multiplexing the Redo Log	13-9
Creating Archived Redo Log Files	13-10
Archiver (ARCn) Process	13-11
Archived Redo Log Files: Naming and Destinations	13-12
Configuring ARCHIVELOG Mode	13-14
Quiz	13-15
Summary	13-16
Practice: Overview	13-17

14 Performing Database Backups

Objectives	14-2
Backup Solutions: Overview	14-3
Oracle Secure Backup	14-4
User-Managed Backup	14-5
Understanding Backup Terminology	14-6
Understanding Types of Backups	14-7
RMAN Backup Types	14-8
Using Recovery Manager (RMAN)	14-10
Configuring Backup Settings	14-11

Oracle-Suggested Backup	14-13
Selecting a Backup Strategy	14-14
Backing Up the Control File to a Trace File	14-15
Managing Backups	14-16
Using RMAN Commands to Create Backups	14-17
Quiz	14-18
Summary	14-19
Practice: Overview	14-20

15 Performing Database Recovery

Objectives	15-2
Opening a Database	15-3
Keeping a Database Open	15-5
Data Recovery Advisor	15-6
Loss of a Control File	15-8
Loss of a Redo Log File	15-9
Loss of a Data File in NOARCHIVELOG Mode	15-11
Loss of a Noncritical Data File in ARCHIVELOG Mode	15-12
Loss of a System-Critical Data File in ARCHIVELOG Mode	15-13
Quiz	15-14
Summary	15-15
Practice: Overview	15-16

16 Moving Data

Objectives	16-2
Moving Data: General Architecture	16-3
Oracle Data Pump: Overview	16-4
Oracle Data Pump: Benefits	16-5
Directory Objects for Data Pump	16-7
Creating Directory Objects	16-8
Data Pump Export and Import Clients: Overview	16-9
Data Pump Utility: Interfaces and Modes	16-10
Performing a Data Pump Export by Using Enterprise Manager Cloud Control	16-11
Performing a Data Pump Import	16-12
Data Pump Import: Transformations	16-13
Using Enterprise Manager Cloud Control to Monitor Data Pump Jobs	16-14
SQL*Loader: Overview	16-15
SQL*Loader Control File	16-17
Loading Methods	16-19
Loading Data by Using Enterprise Manager Cloud Control	16-20
SQL*Loader Express Mode	16-21

External Tables	16-23
External Table: Benefits	16-24
Defining an External Tables with ORACLE_LOADER	16-25
External Table Population with ORACLE_DATAPUMP	16-26
Using External Tables	16-27
Data Dictionary	16-28
Quiz	16-29
Summary	16-31
Practice: Overview	16-32

17 Database Maintenance

Objectives	17-2
Database Maintenance	17-3
Viewing the Alert History	17-4
Terminology	17-5
Automatic Workload Repository (AWR): Overview	17-6
AWR Infrastructure	17-7
Automatic Workload Repository	17-8
AWR Baselines	17-10
Accessing the AWR Page	17-11
Managing the AWR	17-12
Statistic Levels	17-13
Automatic Database Diagnostic Monitor (ADDM)	17-14
ADDM Findings in Enterprise Manager Cloud Control	17-15
ADDM Findings in Enterprise Manager Database Express	17-16
Advisory Framework	17-17
Viewing the Advisor Central Page in Enterprise Manager Cloud Control	17-19
Using Packages to Invoke the Advisors	17-20
Automated Maintenance Tasks	17-21
Automated Maintenance Tasks Configuration	17-23
Server-Generated Alerts	17-24
Setting Metrics Thresholds	17-25
Reacting to Alerts	17-26
Alert Types and Clearing Alerts	17-27
Quiz	17-28
Summary	17-29
Practice: Overview	17-30

18 Managing Performance

- Objectives 18-2
- Performance Monitoring 18-3
- Tuning Activities 18-5
- Performance Planning 18-6
- Instance Tuning 18-8
- Performance Tuning Methodology 18-9
- Performance Tuning Data 18-10
- Using the Enterprise Manager Database Express Performance Hub Page 18-11
- Using the Enterprise Manager Cloud Control Performance Home Page 18-13
- Monitoring Session Performance 18-14
- Performance Monitoring: Top Sessions 18-15
- Displaying Session-Related Statistics 18-16
- Performance Monitoring: Top Services 18-17
- Displaying Service-Related Statistics 18-18
- Viewing Wait Events 18-19
- Oracle Wait Events 18-20
- Memory Management: Overview 18-21
- Managing Memory Components 18-22
- Efficient Memory Usage: Guidelines 18-23
- Automatic Memory Management: Overview 18-25
- Oracle Database Memory Parameters 18-26
- Enabling Automatic Memory Management (AMM) by Using Enterprise Manager Cloud Control 18-27
- Monitoring Automatic Memory Management 18-28
- Automatic Shared Memory Management: Overview 18-30
- Enabling Automatic Shared Memory Management (ASMM) 18-31
- Understanding Automatic Shared Memory Management 18-32
- Automatic Shared Memory Advisor 18-33
- Enabling Automatic Shared Memory Management 18-34
- Disabling Automatic Shared Memory Management 18-35
- Using V\$PARAMETER to View Memory Component Sizes 18-36
- Managing the Program Global Area (PGA) 18-37
- Dynamic Performance Statistics 18-39
- Troubleshooting and Tuning Views 18-41
- Quiz 18-42
- Summary 18-44
- Practice: Overview 18-45

19 Managing Performance: SQL Tuning

- Objectives 19-2
- SQL Tuning 19-3
- Oracle Optimizer: Overview 19-4
- Optimizer Statistics 19-5
- Optimizer Statistics Collection 19-6
- Using the Optimizer Statistics Console 19-8
- Setting Global Preferences by Using Enterprise Manager Cloud Control 19-9
- Gathering Optimizer Statistics Manually 19-10
- Setting Optimizer Statistics Preferences 19-12
- Concurrent Statistics Gathering 19-14
- Viewing Statistics Information 19-15
- SQL Plan Directives 19-17
- Adaptive Execution Plans 19-18
- Using the SQL Advisors 19-19
- Automatic SQL Tuning Results 19-20
- Implementing Automatic Tuning Recommendations 19-21
- SQL Tuning Advisor: Overview 19-22
- Using the SQL Tuning Advisor 19-23
- SQL Tuning Advisor Recommendations 19-25
- Duplicate SQL 19-26
- SQL Access Advisor: Overview 19-27
- Using the SQL Access Advisor 19-28
- Workload Source 19-29
- Recommendation Options 19-30
- Reviewing Recommendations 19-32
- SQL Performance Analyzer: Overview 19-33
- SQL Performance Analyzer: Use Cases 19-34
- Using SQL Performance Analyzer 19-35
- Quiz 19-36
- Summary 19-40
- Practice: Overview 19-41

20 Using Database Resource Manager

- Objectives 20-2
- Database Resource Manager: Overview 20-3
- Database Resource Manager: Concepts 20-4
- Using the Resource Manager 20-5
- Default Plan for Maintenance Windows 20-7
- Default Plan 20-8
- Creating a Simple Resource Plan 20-9

Creating a Complex Resource Plan	20-10
Specifying Resource Plan Directives	20-12
Resource Allocation Methods for Resource Plans	20-13
Comparison of EMPHASIS and RATIO	20-14
Active Session Pool Mechanism	20-16
Specifying Thresholds	20-17
Setting Idle Timeouts	20-19
Limiting CPU Utilization at the Database Level	20-20
Limiting CPU Utilization at the Server Level: Instance Caging	20-22
Instance Caging: Examples	20-23
Monitoring Instance Caging	20-24
Runaway Queries and Resource Manager	20-25
Resource Consumer Group Mapping	20-27
Activating a Resource Plan	20-29
Database Resource Manager Information	20-30
Viewing Resource Manager Statistics	20-31
Monitoring the Resource Manager	20-32
Quiz	20-34
Summary	20-35
Practice: Overview	20-36

21 Using Oracle Scheduler to Automate Tasks

Objectives	21-2
Simplifying Management Tasks	21-3
Understanding a Simple Job	21-4
Oracle Scheduler Core Components	21-5
Using Oracle Scheduler	21-6
Quiz	21-8
Persistent Lightweight Jobs	21-9
Using a Time-Based or Event-Based Schedule	21-10
Creating a Time-Based Job	21-11
Creating an Event-Based Schedule	21-13
Creating Event-Based Schedules by Using Enterprise Manager Cloud Control	21-14
Creating an Event-Based Job	21-15
Event-Based Scheduling	21-16
Creating Complex Schedules	21-18
Quiz	21-19
Using Email Notification	21-20
Adding and Removing Email Notifications	21-21
Creating Job Chains	21-22

Example of a Chain	21-24
Using Advanced Scheduler Features	21-25
Job Classes	21-26
Windows	21-28
Prioritizing Jobs Within a Window	21-29
Creating a Job Array	21-30
Quiz	21-32
Creating a File Watcher and an Event-Based Job	21-33
Enabling File Arrival Events from Remote Systems	21-35
Scheduling Remote Database Jobs	21-36
Creating Remote Database Jobs	21-37
Scheduling Multiple Destination Jobs	21-38
Viewing Scheduler Meta Data	21-39
Quiz	21-41
Summary	21-42
Practice: Overview	21-43

Appendix A: Working with Oracle Support

Objectives	A-2
Working with Oracle Support	A-3
Using My Oracle Support	A-4
Researching an Issue	A-6
Logging Service Requests	A-8
Accessing My Oracle Support Community	A-10
Managing Patches	A-11
Applying a Patch Release	A-12
Enterprise Manager Cloud Control: My Oracle Support Integration	A-13
Using the Patch Advisor	A-14
Online Patching: Overview	A-15
Installing an Online Patch	A-16
Benefits of Online Patching	A-17
Conventional Patching and Online Patching	A-18
Online Patching Considerations	A-19
Quiz	A-20
Using the Support Workbench	A-21
Accessing the Support Workbench	A-22
Viewing Problem Details	A-23
Viewing Incident Details	A-24
Packaging and Uploading Diagnostic Data to Oracle Support	A-25
Tracking the Service Request and Implementing Repairs	A-26
Summary	A-27

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

Oracle University and Error : You are not a Valid Partner use only

1

Introduction



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Explain the course objectives
- Describe the course schedule
- Describe the evolution of Oracle Database
- Describe Enterprise Cloud Computing
- Describe the HR schema



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Course Objectives

After completing this course, you should be able to:

- Describe Oracle Database architecture
- Configure the database to support your applications
- Manage database security and implement auditing
- Implement basic backup and recovery procedures
- Move data between databases and files
- Employ basic monitoring procedures and manage performance
- Manage resources and automate tasks
- Work with Oracle Support



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this course, you learn how to administer Oracle Database 12c Release 1.

You also configure the database to support an application and perform tasks such as creating users, defining storage structures, and setting up security. This course uses a fictional application. However, you perform all the core tasks that are necessary for a real application.

Database administration does not end after you configure your database. You also learn the basics of protecting it by designing a backup and recovery strategy. In addition you learn how to monitor the database to ensure that it operates smoothly.

Suggested Schedule

Day	Lessons	Day	Lessons
1	1. Introduction 2. Exploring the Oracle Database Architecture 3. Oracle Database Management Tools 4. Managing the Database Instance	3	10. Managing Data Concurrency 11. Implementing Oracle Database Auditing 12. Backup and Recovery Concepts 13. Backup and Recovery: Configuration
2	5. Configuring the Oracle Network Environment 6. Administering User Security 7. Managing Database Storage Structures 8. Managing Space 9. Managing Undo Data	4	14. Performing Database Backups 15. Performing Database Recovery 16. Moving Data 17. Performing Database Maintenance
		5	18. Managing Performance 19. Managing Performance: SQL Tuning 20. Using Resource Manager 21. Using Oracle Scheduler



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Database Innovation

... continuing with

Oracle Database 12c

... with Oracle Database 11g

... with Oracle
Database 10g

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

Audit Vault

Database Vault

Secure Enterprise Search

Grid Computing

Automatic Storage Mgmt

Self Managing Database

ORACLE

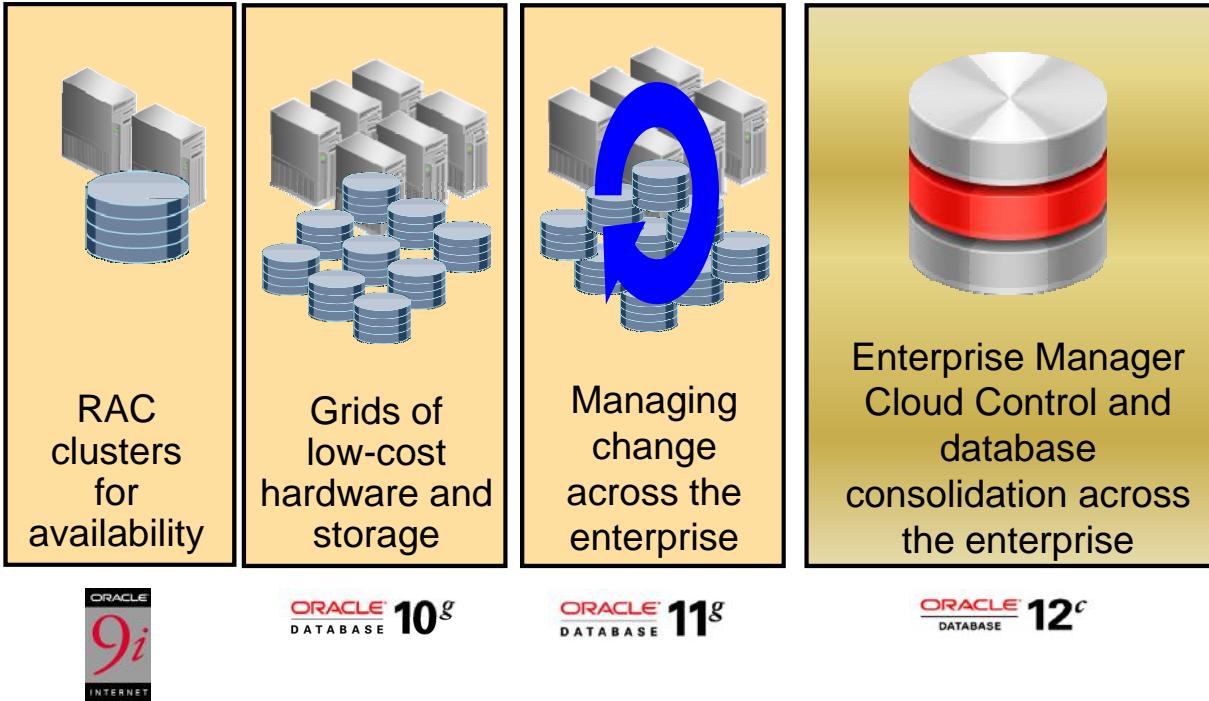
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 including 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 © 2014, Oracle and/or its affiliates. All rights reserved.



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

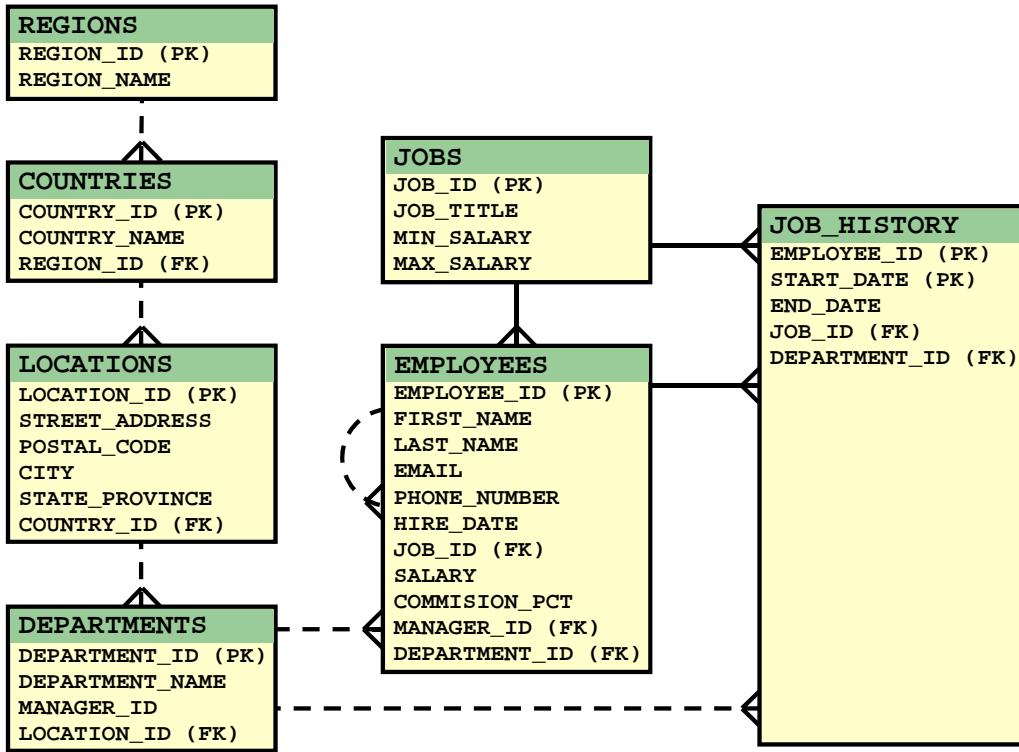
Oracle Database 11g consolidates and extends Oracle's unique ability to deliver the benefits of grid computing, transforming data centers from silos of isolated system resources to shared pools of servers and storage.

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 server 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.
- A single console manages the entire Cloud life cycle—plan, set up, deliver, and 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.

Course Examples: HR Sample Schema



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The examples used in this course are from a human resources (HR) schema, which can be created as part of the starter database.

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

- Each department may be the employer of one or more employees. Each employee may be assigned to only one department.
- Each job must be a job for one or more employees. Each employee must be currently assigned to only one job.
- When an employee changes his or her department or job, a record in the `JOB_HISTORY` table records the start and end dates of the past assignments.
- `JOB_HISTORY` records are identified by a composite primary key (PK): the `EMPLOYEE_ID` and the `START_DATE` columns.

Notation: PK = Primary Key, FK = Foreign Key

Solid lines represent mandatory foreign key (FK) constraints and dashed lines represent optional FK constraints.

The `EMPLOYEES` table also has an FK constraint with itself. This is an implementation of the business rule: Each employee may be reporting directly to only one manager. The FK is optional because the top employee does not report to another employee.

Summary

In this lesson, you should have learned how to:

- Explain the course objectives
- Describe the course schedule
- Describe the evolution of Oracle Database
- Describe Enterprise Cloud Computing
- Describe the HR schema



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Exploring Oracle Database Architecture



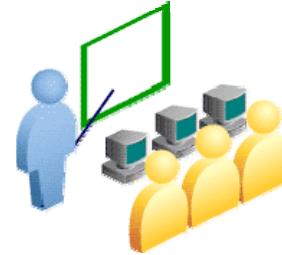
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- List the major architectural components of Oracle Database
- Explain memory structures
- Describe background processes
- Correlate logical and physical storage structures
- Describe pluggable databases
- Describe ASM storage components

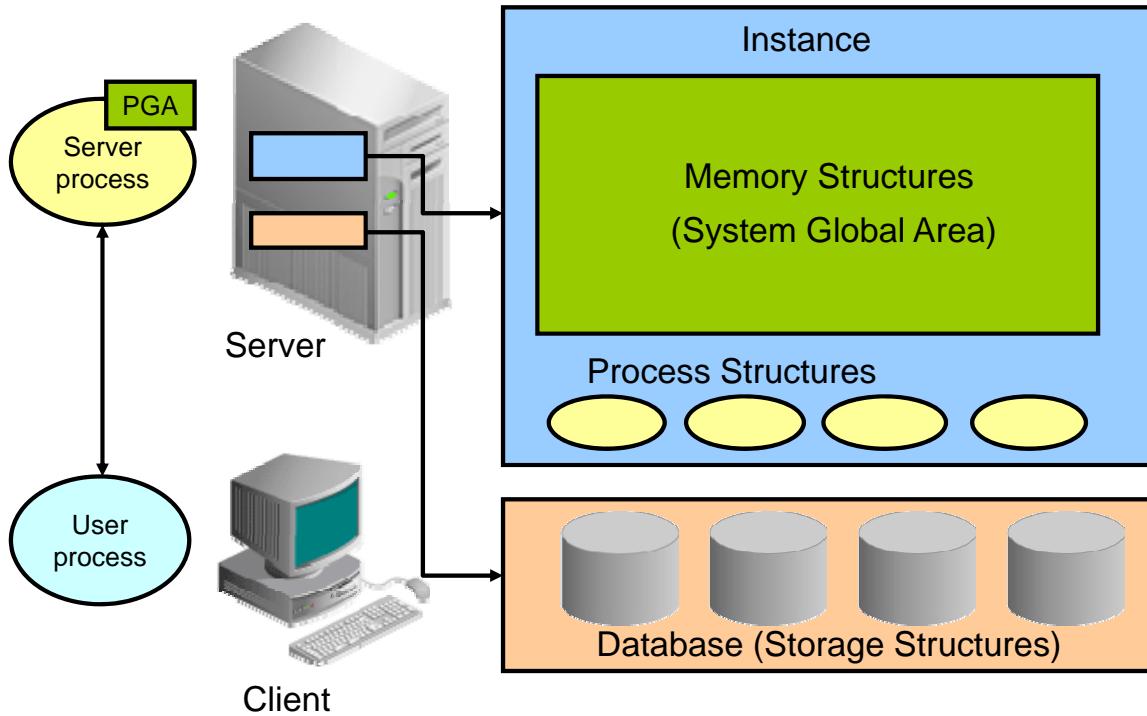


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This lesson provides a detailed overview of Oracle Database architecture. You learn about physical and logical structures and about various components.

Oracle Database Server Architecture: Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

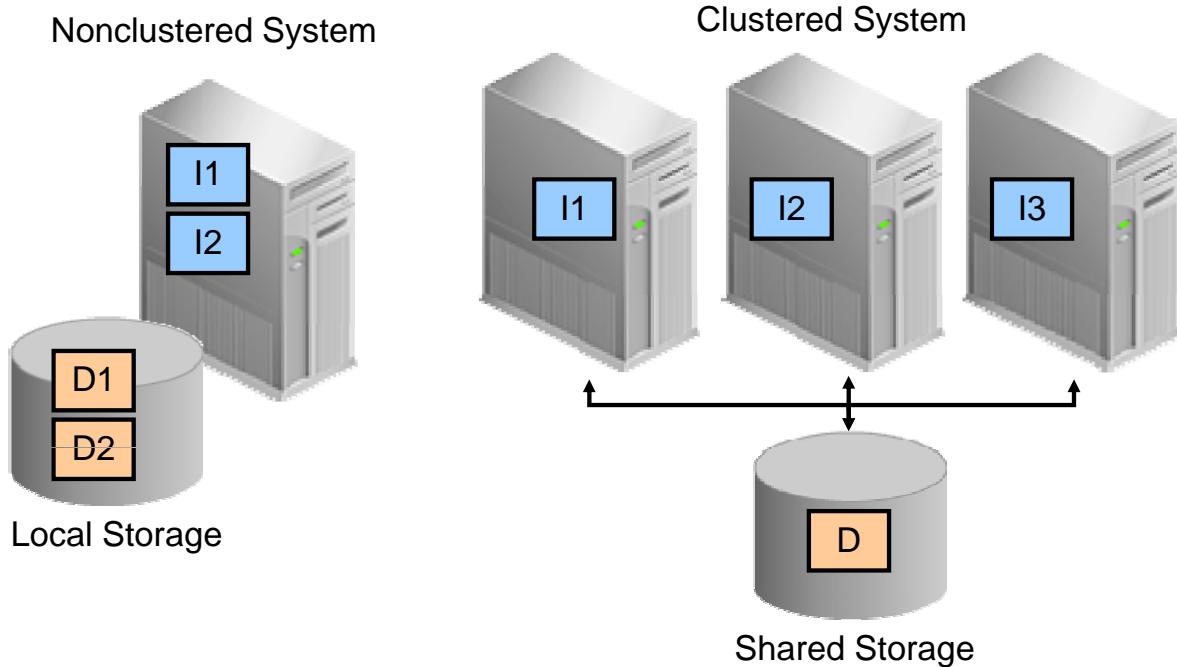
There are three major structures in Oracle Database server architecture: memory structures, process structures, and storage structures. A basic Oracle database system consists of an Oracle database and a database instance.

The database consists of both physical structures and logical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting access to logical storage structures.

The instance consists of memory structures and background processes associated with that instance. Every time an instance is started, a shared memory area called the System Global Area (SGA) is allocated and the background processes are started. Processes are jobs that work in the memory of computers. A process is defined as a “thread of control” or a mechanism in an operating system that can run a series of steps. After starting a database instance, the Oracle software associates the instance with a specific database. This is called *mounting the database*. The database is then ready to be opened, which makes it accessible to authorized users.

Note: Oracle Automatic Storage Management (ASM) uses the concept of an instance for the memory and process components, but is not associated with a specific database.

Oracle Database Instance Configurations



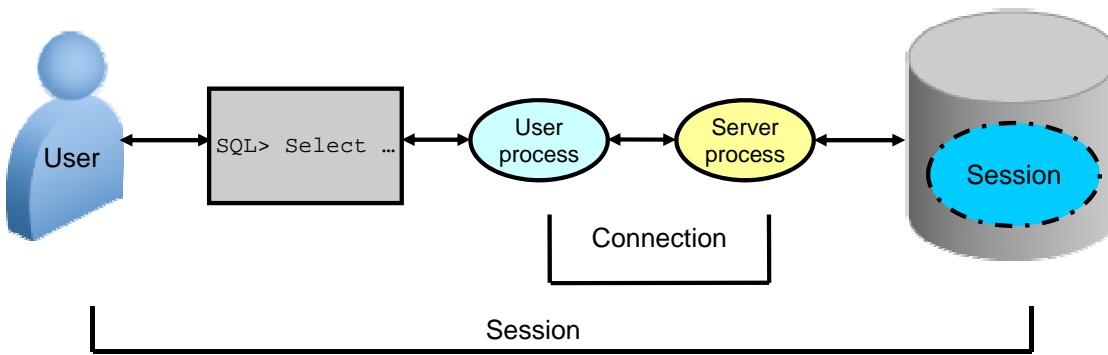
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each database instance is associated with one and only one database. If there are multiple databases on the same server, then there is a separate and distinct database instance for each database. A database instance cannot be shared. A Real Applications Cluster (RAC) database usually has multiple instances on separate servers for the same shared database. In this model, the same database is associated with each RAC instance, which meets the requirement that, at most, only one database is associated with an instance.

Connecting to the Database Instance

- Connection: Communication between a user process and an instance
- Session: Specific connection of a user to an instance through a user process



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

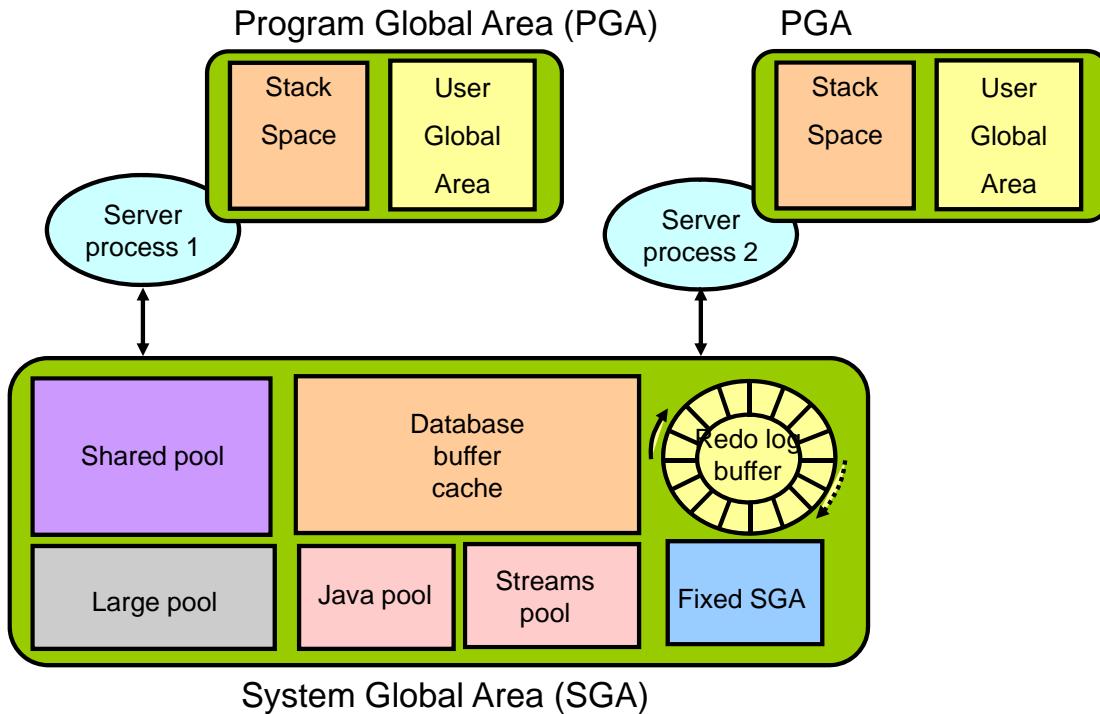
Connections and sessions are closely related to user processes but are very different in meaning.

A *connection* is a communication pathway between a user process and an Oracle Database instance. A communication pathway is established using available interprocess communication mechanisms (on a computer that runs both the user process and Oracle Database) or network software (when different computers run the database application and Oracle Database, and communicate through a network).

A *session* represents the state of a current user login to the database instance. For example, when a user starts SQL*Plus, the user must provide a valid username and password, and then a session is established for that user. A session lasts from the time a user connects until the user disconnects or exits the database application.

Multiple sessions can be created and exist concurrently for a single Oracle database user using the same username. For example, a user with the username/password of HR/HR can connect to the same Oracle Database instance several times.

Oracle Database Memory Structures



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Database creates and uses memory structures for various purposes. For example, memory stores program code being run, data that is shared among users, and private data areas for each connected user.

Two basic memory structures are associated with an instance:

- **System Global Area (SGA):** Group of shared memory structures, known as SGA components, which contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.
- **Program Global Areas (PGA):** Memory regions that contain data and control information for a server or background process. A PGA is nonshared memory created by Oracle Database when a server or background process is started. Access to the PGA is exclusive to the server process. Each server process and background process has its own PGA.

The SGA is the memory area that contains data and control information for the instance. The SGA includes the following data structures:

- **Shared pool:** Caches various constructs that can be shared among users
- **Database buffer cache:** Caches blocks of data retrieved from the database
- **Redo log buffer:** Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on the disk
- **Large pool:** Optional area that provides large memory allocations for certain large processes, such as Oracle backup and recovery operations, and I/O server processes
- **Java pool:** Used for all session-specific Java code and data in the Java Virtual Machine (JVM)
- **Streams pool:** Used by Oracle Streams to store information required by capture and apply
- **Fixed SGA:** An internal housekeeping area containing general information about the state of the database and the instance, and information communicated between processes

When you start the instance, the amount of memory allocated for the SGA is displayed.

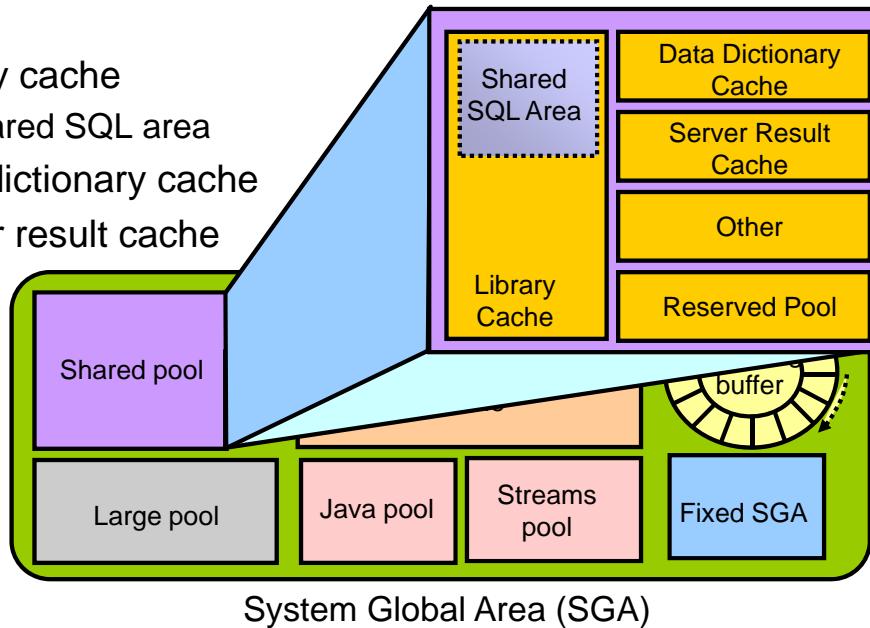
A Program Global Area (PGA) is a memory region that contains data and control information for each server process. An Oracle server process services a client's requests. Each server process has its own private PGA that is allocated when the server process is started. Access to the PGA is exclusive to that server process, and the PGA is read and written only by the Oracle code acting on its behalf. The PGA is divided into two major areas: stack space and the user global area (UGA).

With the dynamic SGA infrastructure, the sizes of the database buffer cache, the shared pool, the large pool, the Java pool, and the Streams pool can change without shutting down the instance.

The Oracle Database server uses initialization parameters to create and manage memory structures. The simplest way to manage memory is to allow the database to automatically manage and tune it for you. To do so (on most platforms), you only have to set a target memory size initialization parameter (`MEMORY_TARGET`) and a maximum memory size initialization parameter (`MEMORY_MAX_TARGET`).

Shared Pool

- Is a portion of the SGA
- Contains:
 - Library cache
 - Shared SQL area
 - Data dictionary cache
 - Server result cache



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The shared pool portion of the SGA contains the library cache, the data dictionary cache, the server result cache containing the SQL query result cache and the PL/SQL function result cache, buffers for parallel execution messages, and control structures.

The *data dictionary* is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database.

The data dictionary is accessed so often by Oracle Database that two special locations in memory are designated to hold dictionary data. One area is called the *data dictionary cache*, also known as the row cache because it holds data as rows instead of buffers (buffers hold entire blocks of data). The other area in memory that holds dictionary data is the *library cache*. All Oracle Database user processes share these two caches for access to data dictionary information.

Oracle Database represents each SQL statement that it runs with a shared SQL area (as well as a private SQL area kept in the PGA). Oracle Database recognizes when two users are executing the same SQL statement and reuses the shared SQL area for those users.

A shared SQL area contains the parse tree and execution plan for a given SQL statement. Oracle Database saves memory by using one shared SQL area for SQL statements run multiple times, which often happens when many users run the same application.

When a new SQL statement is parsed, Oracle Database allocates memory from the shared pool to store in the shared SQL area. The size of this memory depends on the complexity of the statement.

Oracle Database processes PL/SQL program units (procedures, functions, packages, anonymous blocks, and database triggers) in much the same way it processes individual SQL statements. Oracle Database allocates a shared area to hold the parsed, compiled form of a program unit. Oracle Database allocates a private area to hold values specific to the session that runs the program unit, including local, global, and package variables (also known as package instantiation) and buffers for executing SQL. If more than one user runs the same program unit, then a single, shared area is used by all users, while all users maintain separate copies of their own private SQL areas, holding values specific to their own sessions.

Individual SQL statements contained in a PL/SQL program unit are processed just like other SQL statements. Despite their origins in a PL/SQL program unit, these SQL statements use a shared area to hold their parsed representations and a private area for each session that runs the statement.

The *server result cache* contains the *SQL query result cache* and *PL/SQL function result cache*, which share the same infrastructure. The server result cache contains result sets, not data blocks.

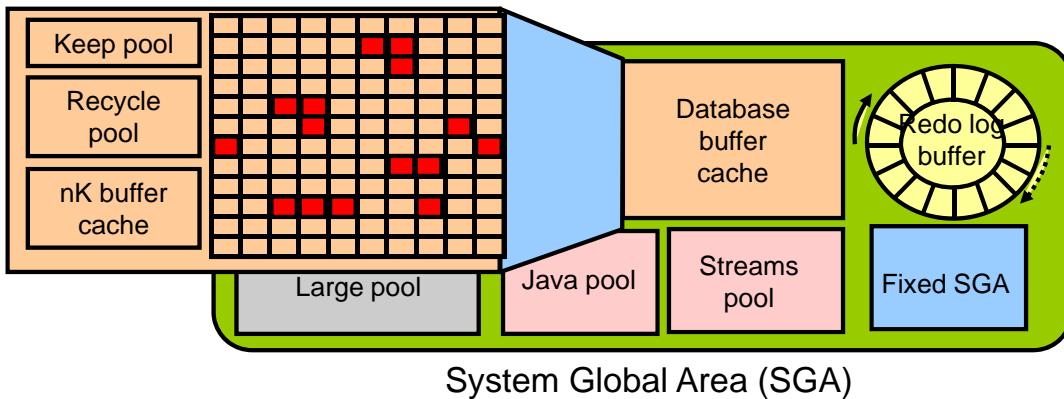
Results of queries and query fragments can be cached in memory in the SQL query result cache. The database server can then use cached results to answer future executions of these queries and query fragments. Because retrieving results from the SQL query result cache is faster than rerunning a query, frequently run queries experience a significant performance improvement when their results are cached.

A PL/SQL function is sometimes used to return the result of a computation whose inputs are one or several parameterized queries issued by the function. In some cases, these queries access data that changes very infrequently compared to the frequency of calling the function. You can include syntax in the source text of a PL/SQL function to request that its results be cached in the PL/SQL function result cache and (to ensure correctness) that the cache be purged when tables in a list of tables experience data manipulation language (DML).

The *reserved pool* is a memory area in the shared pool that Oracle Database can use to allocate large contiguous chunks of memory.

Database Buffer Cache

- Is part of the SGA
- Holds copies of data blocks that are read from data files
- Is shared by all concurrent users



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The database buffer cache is the portion of the SGA that holds block images read from the data files or constructed dynamically to satisfy the read consistency model. All users who are concurrently connected to the instance share access to the database buffer cache.

The first time an Oracle Database user process requires a particular piece of data, it searches for the data in the database buffer cache. If the process finds the data already in the cache (a cache hit), it can read the data directly from memory. If the process cannot find the data in the cache (a cache miss), it must copy the data block from a data file on disk into a buffer in the cache before accessing the data. Accessing data through a cache hit is faster than accessing data through a cache miss.

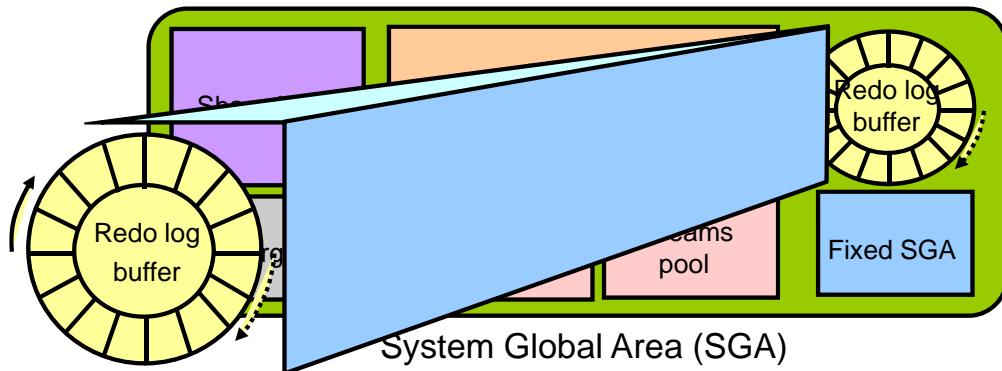
The buffers in the cache are managed by a complex algorithm that uses a combination of least recently used (LRU) lists and touch count. The LRU helps to ensure that the most recently used blocks tend to stay in memory to minimize disk access.

The keep buffer pool and the recycle buffer pool are used for specialized buffer pool tuning. The keep buffer pool is designed to retain buffers in memory longer than the LRU would normally retain them. The recycle buffer pool is designed to flush buffers from memory faster than the LRU normally would.

Additional buffer caches can be configured to hold blocks of a size that is different from the default block size.

Redo Log Buffer

- Is a circular buffer in the SGA
- Holds information about changes made to the database
- Contains redo entries that have the information to redo changes made by operations such as DML and DDL



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

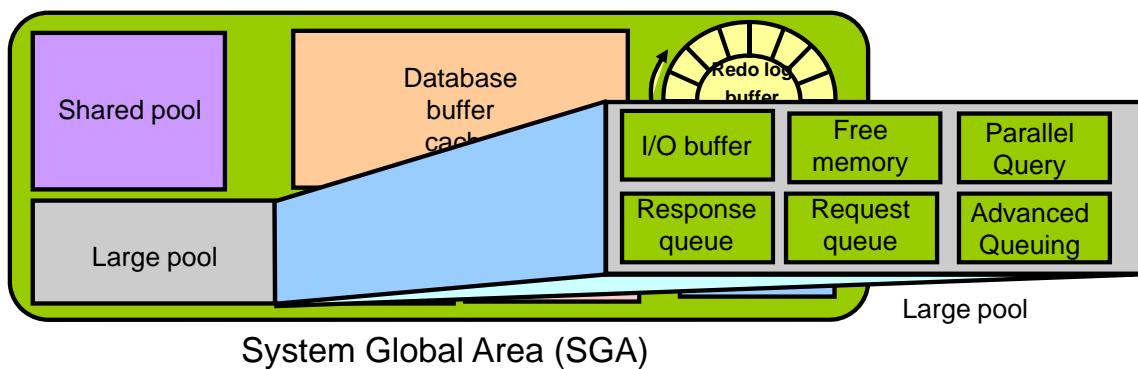
The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct (or redo) changes that are made to the database by DML, DDL, or internal operations. Redo entries are used for database recovery if necessary.

As the server process makes changes to the buffer cache, redo entries are generated and written to the redo log buffer in the SGA. The redo entries take up continuous, sequential space in the buffer. The log writer background process writes the redo log buffer to the active redo log file (or group of files) on disk.

Large Pool

Provides large memory allocations for:

- Session memory for the shared server and the Oracle XA interface
- I/O server processes
- Oracle Database backup and restore operations



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The database administrator can configure an optional memory area called the *large pool* to provide large memory allocations for:

- Session memory for the shared server and the Oracle XA interface (used where transactions interact with multiple databases)
- I/O server processes
- Oracle Database backup and restore operations
- Parallel Query operations
- Advanced Queuing memory table storage

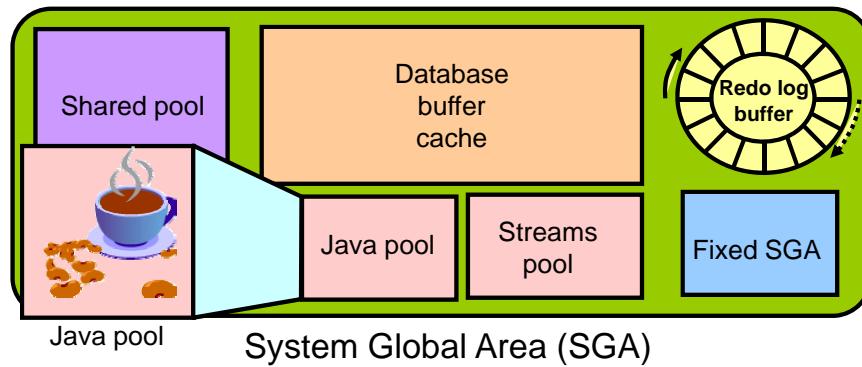
By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle Database can use the shared pool primarily for caching shared SQL and avoid the performance overhead that is caused by shrinking the shared SQL cache.

In addition, the memory for Oracle Database backup and restore operations, for I/O server processes, and for parallel buffers is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such large memory requests than the shared pool.

The large pool is not managed by a least recently used (LRU) list.

Java Pool

Java pool memory is used to store all session-specific Java code and data in the JVM.



ORACLE

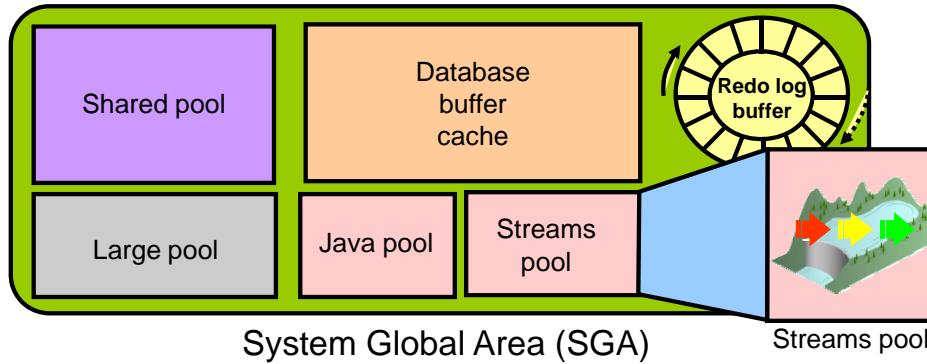
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Java pool memory is used to store all session-specific Java code and data in the Java Virtual Machine (JVM). Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

Streams Pool

Streams pool memory is used exclusively by Oracle Streams to:

- Store buffered queue messages
- Provide memory for Oracle Streams processes



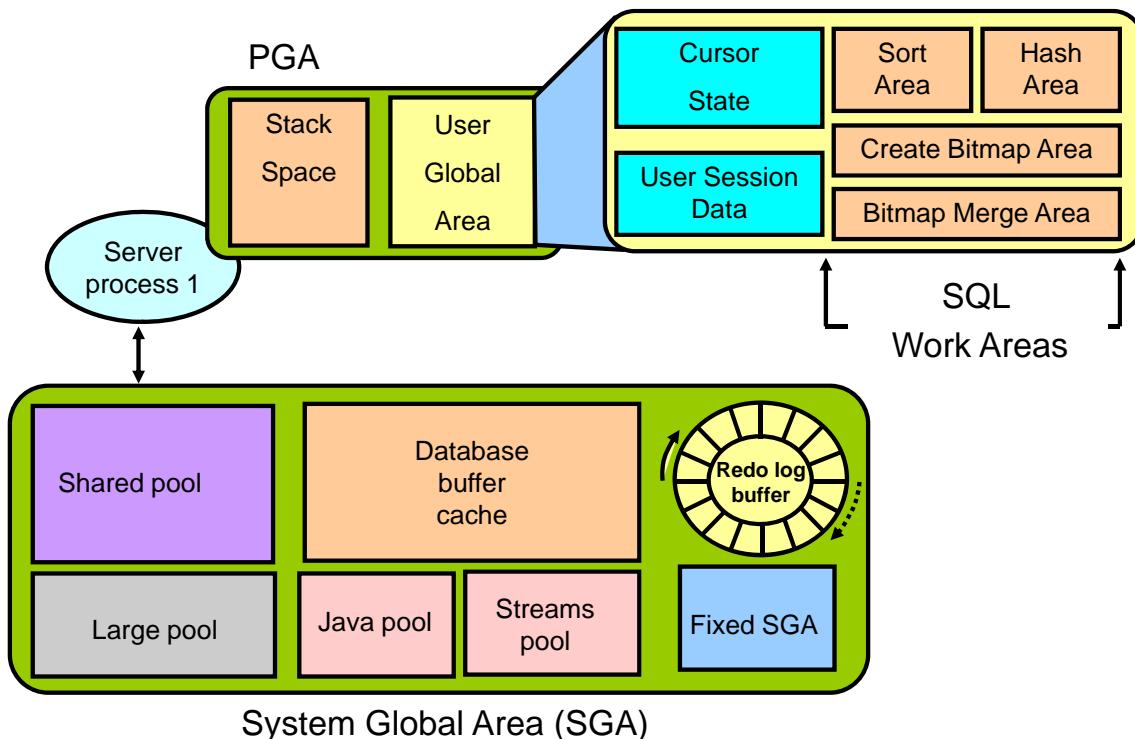
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Streams pool is used exclusively by Oracle Streams. The Streams pool stores buffered queue messages, and it provides memory for Oracle Streams capture processes and apply processes.

Unless you specifically configure it, the size of the Streams pool starts at zero. The pool size grows dynamically as needed when Oracle Streams is used.

Program Global Area (PGA)



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Program Global Area (PGA) is a private memory region containing data and control information for a server process. Each server process has a distinct PGA. Access to it is exclusive to that server process and it is read only by Oracle code acting on behalf of it. It is not available for developer's code.

Every PGA contains stack space. In a dedicated server environment, each user connecting to the database instance has a separate server process. For this type of connection, the PGA contains a subdivision of memory known as the user global area (UGA). The UGA is composed of the following:

- Cursor area for storing runtime information on cursors
- User session data storage area for control information about a session
- SQL working areas for processing SQL statements consisting of:
 - A sort area for functions that order data such as ORDER BY and GROUP BY
 - A hash area for performing hash joins of tables
 - A create bitmap area used in bitmap index creation common to data warehouses
 - A bitmap merge area used for resolving bitmap index plan execution

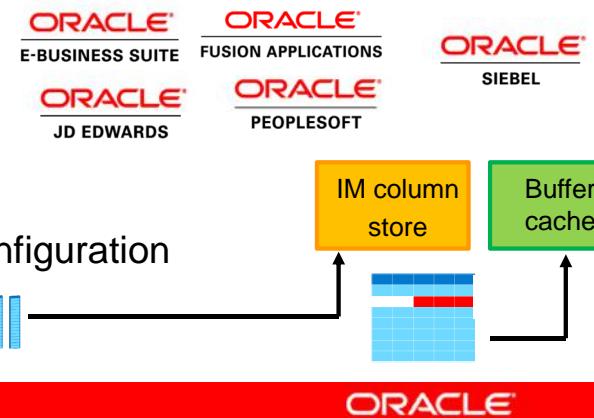
In a shared server environment, multiple client users share the server process. In this model, the UGA is moved into the SGA (shared pool or large pool if configured) leaving the PGA with only stack space.

In-Memory Column Store: Introduction

- Instant query response:
 - Faster queries on very large tables on any columns (100x)
 - Use of scans, joins, and aggregates
 - Without indexes
 - Best suited for analytics: few columns, many rows
- Faster DML: Removal of most analytics indexes (3 to 4x)

- Full application transparency

- Easy setup:
 - In-memory column store configuration
 - Segment attributes



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The In-Memory Column Store feature enables objects (tables, partitions, and other types) to be stored in memory in a new format known as the *columnar format*. This format enables scans, joins, and aggregates to perform much faster than the traditional on-disk format, thus providing fast reporting and DML performance for both OLTP and DW environments. This is particularly useful for analytic applications that operate on few columns returning many rows rather than for OLTP that operates on few rows returning many columns. The DBA must define the segments that are to be populated into the in-memory column store (IM column store), such as hot tables, partitions, and more precisely the more frequently accessed columns.

The in-memory columnar format does not replace the on-disk or buffer cache format. It is a consistent copy of a table or of some columns of a table converted to the new columnar format that is independent of the disk format and only available in memory. Because of this independence, applications are able to transparently use this option without any changes. For the data to be converted into the new columnar format, a new pool is requested in the SGA. The pool is the IM column store.

If sufficient space is allocated for the IM column store, a query that accesses objects that are candidates to be populated into the IM column store performs much faster. The improved performance allows ad hoc analytic queries to be executed directly on the real-time transaction data without impacting the existing workload.

There are three main advantages:

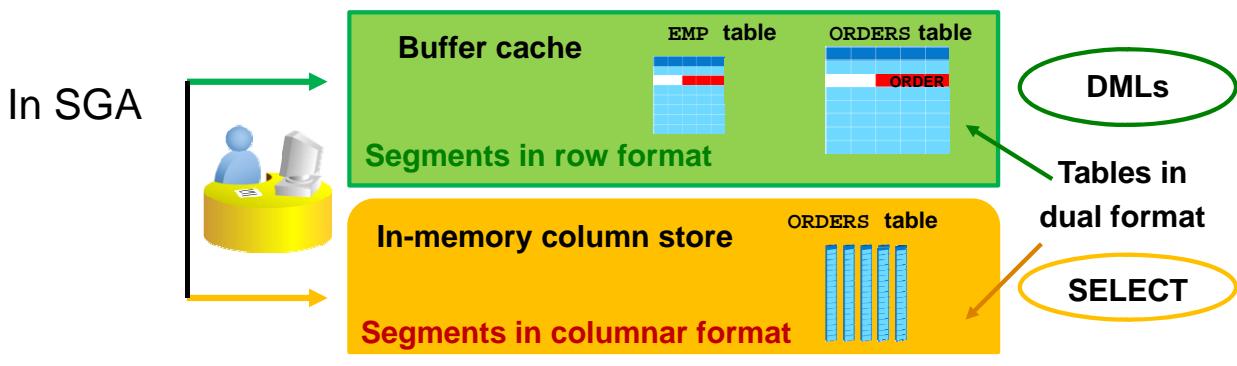
- Queries run a lot faster: All data can be populated in memory in a compressed columnar format. No index is required and used. Queries run at least 100 times faster than when fetching data from the buffer cache, thanks to the columnar compressed format.
- DMLs are faster: Analytics indexes can be eliminated by being replaced by scans of the IM column store representation of the table.
- Arbitrary ad hoc queries run with good performance, because the table behaves as if all columns are indexed.

Note: The In-Memory Column Store feature is included with the Oracle Database In-Memory option.

Refer to the *Oracle Database Administrator's Guide* for detailed information about this feature.

In-Memory Column Store: Overview

- A new pool in the SGA: In-Memory column store
 - Segments populated into the IM column store are converted into a columnar format.
 - In-Memory segments are transactionally consistent with the buffer cache.
- Only one segment on disk and in row format



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The in-memory columnar format does not replace the on-disk or buffer cache format. This means that when a segment such as a table or a partition is populated into the IM column store, the on-disk format segment is automatically converted into a columnar format and optionally compressed. The columnar format is a pure in-memory format. There is no columnar format storage on disk. It never causes additional writes to disk and therefore does not require any logging or undo space.

All data is stored on disk in the traditional row format.

Moreover, the columnar format of a segment is a transaction-consistent copy of the segment either on disk or in the buffer cache. Transaction consistency between the two pools is maintained.

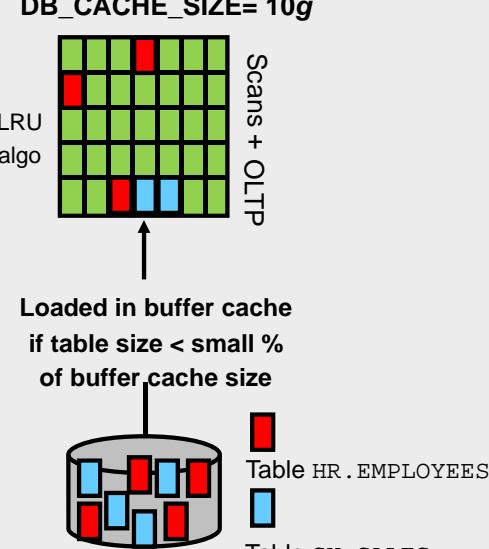
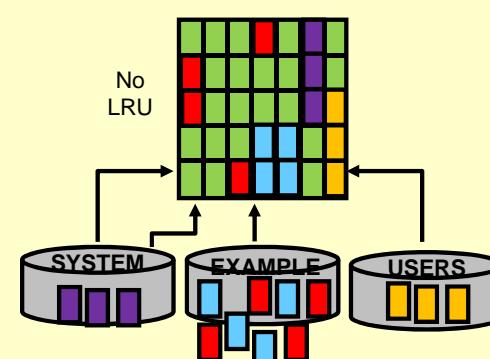
If sufficient space is allocated to the IM column store in SGA, a query that accesses objects that are populated into the IM column store performs much faster. The improved performance allows more ad hoc analytic queries to be executed directly on real-time transaction data without impacting the existing workload. A lack of IM column store space does not prevent statements from executing against tables that could have been populated into the IM column store.

The DBA must decide, according to the types of queries and DMLs that are executed against the segments, which segments should be defined as non in-memory segments and which should be defined as in-memory segments. The DBA can also define more precisely which columns are good candidates for IM column store:

- In **row format exclusively**: The segments that are being frequently accessed by OLTP-style queries, which operate on few rows returning many columns, are good candidates for the buffer cache. These segments should not necessarily be defined as in-memory segments and should be sent to the buffer cache only.
- In **dual format simultaneously**: The segments that are being frequently accessed by analytical-style queries, which operate on many rows returning a few columns, are good candidates for IM column store. If a segment is defined as an in-memory segment but has some columns that are defined as non in-memory columns, the queries that select any non in-memory columns are sent to the buffer cache and those selecting in-memory columns only are sent to the IM column store. Any fetch-by-rowid is performed on the segment through the buffer cache.

Any DML performed on these objects is executed via the buffer cache.

Full Database In-Memory Caching

Traditional Buffer Cache Usage	Full Database In-Memory Caching
<p>DB_CACHE_SIZE= 10g</p>  <p>Scans + OLTP</p> <p>LRU algo</p> <p>Loaded in buffer cache if table size < small % of buffer cache size</p> <p>Table HR. EMPLOYEES</p> <p>Table SH. SALES</p>	<p>Entire database loaded into the buffer cache:</p> <ul style="list-style-type: none"> • Huge performance benefits • Two modes <ul style="list-style-type: none"> – Full Database Caching – Force Full Database Caching 

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The current algorithm for table scans loads a table into the buffer cache only when the table size is less than a small percent of the buffer cache size. For very large tables, the database uses a direct path read, which loads blocks directly into the PGA and bypasses the SGA, to avoid flooding the buffer cache. The DBA must explicitly declare small lookup tables, which are accessed frequently, as CACHE to load data into memory and avoid bypassing the SGA. This clause indicates that the blocks retrieved for these tables are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.

The Full Database In-memory Caching feature enables an entire database to be cached in memory when the database size (sum of all data files, SYSTEM tablespace, LOB CACHE files minus SYSAUX, TEMP) is smaller than the buffer cache size. Caching and running a database from memory leads to huge performance benefits. Two modes can be used:

- **Full Database Caching:** Implicit default and automatic mode in which an internal calculation determines if the database can be fully cached for an instance. NOCACHE LOBs are not cached in Full Database Caching but in Force Full Database Caching mode even NOCACHE LOBs are cached.

- **Force Full Database Caching:** Neither Full Database Caching nor Force Full Database Caching forces or prefetches data into memory. Workload must access the data first for them to be cached. It considers the entire database as eligible to be completely cached in the buffer cache. This mode requires the DBA to execute the `ALTER DATABASE FORCE FULL DATABASE CACHING` command. This mode takes precedence over Full Database Caching mode. To revert to traditional caching, use the `ALTER DATABASE NO FORCE FULL DATABASE CACHING` command.

Refer to the *Oracle Database Administrators Guide* and the *Oracle Database Performance Tuning Guide* for detailed information about this feature.

Quiz

The memory region that contains data and control information for a server or background process is called:

- a. Shared pool
- b. PGA
- c. Buffer cache
- d. User session data



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

What is read into the database buffer cache from data files?

- a. Rows
- b. Changes
- c. Blocks
- d. SQL



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Process Architecture

- User process
 - Is the application or tool that connects to the Oracle database
- Database processes
 - Server process: Connects to the Oracle instance and is started when a user establishes a session
 - Background processes: Are started when an Oracle instance is started
- Daemon / Application processes
 - Networking listeners
 - Grid Infrastructure daemons



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

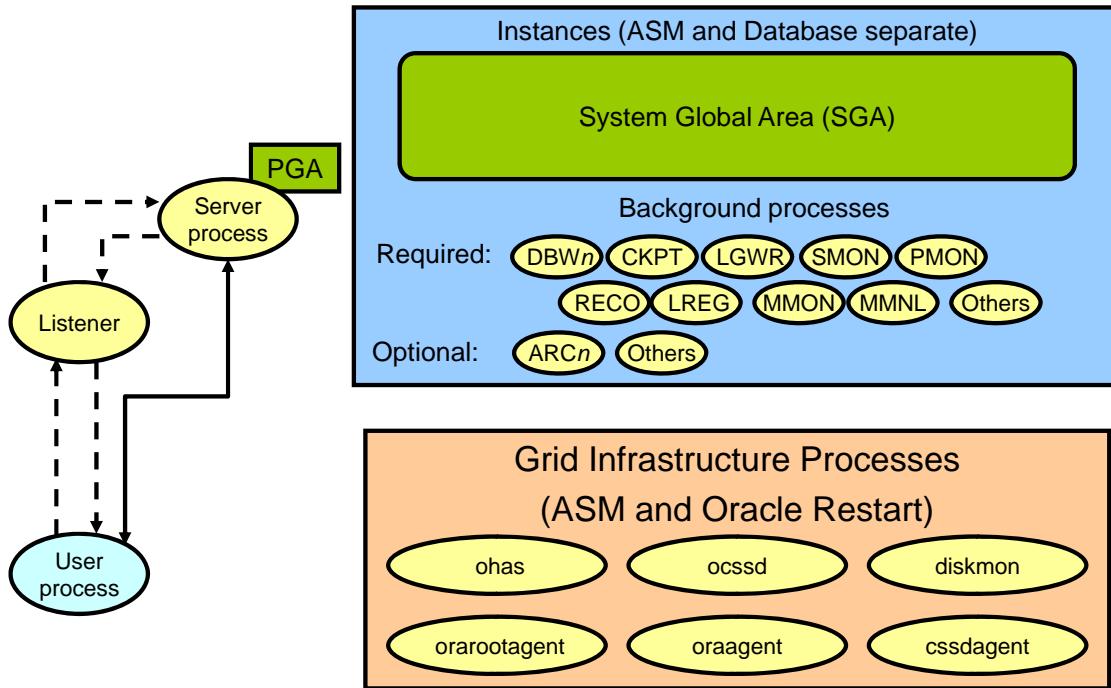
The processes in an Oracle Database system can be divided into three major groups:

- User processes that run the application or Oracle tool code
- Oracle Database processes that run the Oracle Database server code (including server processes and background processes)
- Oracle daemons and application processes not specific to a single database

When a user runs an application program or an Oracle tool such as SQL*Plus, the term *user process* is used to refer to the user's application. The user process may or may not be on the database server machine. Oracle Database also creates a *server process* to execute the commands issued by the user process. In addition, the Oracle server also has a set of *background processes* for an instance that interact with each other and with the operating system to manage the memory structures, asynchronously perform I/O to write data to disk, and perform other required tasks. The process structure varies for different Oracle Database configurations, depending on the operating system and the choice of Oracle Database options. The code for connected users can be configured as a dedicated server or a shared server.

- **Dedicated server:** For each session, the database application is run by a user process that is served by a dedicated server process that executes Oracle database server code.
- **Shared server:** Eliminates the need for a dedicated server process for each connection. A dispatcher directs multiple incoming network session requests to a pool of shared server processes. A shared server process serves any client request.

Process Structures



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Server Processes

Oracle Database creates server processes to handle the requests of user processes connected to the instance. The user process represents the application or tool that connects to the Oracle database. It may be on the same machine as the Oracle database, or it may exist on a remote client and use a network to reach the Oracle database. The user process first communicates with a listener process that creates a server process in a dedicated environment.

Server processes created on behalf of each user's application can perform one or more of the following:

- Parse and run SQL statements issued through the application.
- Read necessary data blocks from data files on disk into the shared database buffers of the SGA (if the blocks are not already present in the SGA).
- Return results in such a way that the application can process the information.

Background Processes

To maximize performance and accommodate many users, a multiprocess Oracle Database system uses some additional Oracle Database processes called *background processes*. An Oracle Database instance can have many background processes.

The background processes commonly seen in non-RAC, non-ASM environments can include the following:

- Database Writer process (DBW n)
- Log Writer process (LGWR)
- Checkpoint process (CKPT)
- System monitor process (SMON)
- Process monitor process (PMON)
- Recoverer process (RECO)
- Listener registration process (LREG)
- Manageability monitor process (MMON)
- Manageability monitor lite process (MMNL)
- Job queue coordinator (CJQ0)
- Job slave processes (Jnnn)
- Archiver processes (ARC n)
- Queue monitor processes (QMNN)

Other background processes may be found in more advanced configurations such as RAC. See the V\$BGPPROCESS view for more information on the background processes.

Some background processes are created automatically when an instance is started, whereas others are started as required.

Other process structures are not specific to a single database, but rather can be shared among many databases on the same server. The Grid Infrastructure and networking processes fall into this category.

Oracle Grid Infrastructure processes on Linux and UNIX systems include the following:

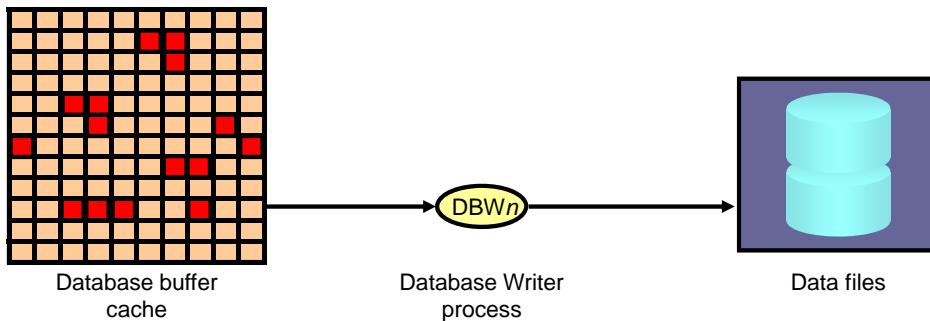
- ohasd (Oracle High Availability Service daemon): Is responsible for starting Oracle Clusterware processes
- ocssd: Cluster Synchronization Service daemon
- diskmon (Disk Monitor daemon): Is responsible for input and output fencing for Oracle Exadata Storage
- cssdagent: Starts, stops, and checks the status of the CSS daemon, ocssd
- oraagent: Extends Clusterware to support Oracle-specific requirements and complex resources
- orarootagent: Is a specialized Oracle agent process that helps manage resources owned by root, such as the network.

Note: For a more detailed list of the background processes, consult the *Oracle Database Reference* guide.

Database Writer Process (DBWn)

Writes modified (dirty) buffers in the database buffer cache to disk:

- Asynchronously while performing other processing
- To advance the checkpoint



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Database Writer process (DBW n) writes the contents of buffers to data files. The DBW n processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk. Although one Database Writer process (DBW0) is adequate for most systems, you can configure additional processes to improve write performance if your system modifies data heavily. The additional processes are named DBW1 through DBW9, DBWa through DBWz, and BW36-BW99. These additional DBW n processes are not useful on uniprocessor systems.

When a buffer in the database buffer cache is modified, it is marked dirty and is added to the head of the checkpoint queue that is kept in system change number (SCN) order. This order therefore matches the order of redo that is written to the redo logs for these changed buffers. When the number of available buffers in the buffer cache falls below an internal threshold (to the extent that server processes find it difficult to obtain available buffers), DBW n writes non-frequently used buffers to the data files from the tail of the LRU list so that processes can replace buffers when they need them. DBW n also writes from the tail of the checkpoint queue to keep the checkpoint advancing.

The SGA contains a memory structure that has the redo byte address (RBA) of the position in the redo stream where recovery should begin in case of an instance failure. This structure acts as a pointer into the redo and is written to the control file by the CKPT process once every three seconds. Because the DBW n writes dirty buffers in SCN order, and because the redo is in SCN order, every time DBW n writes dirty buffers from the LRU list, it also advances the pointer held in the SGA memory structure so that instance recovery (if required) begins reading the redo from approximately the correct location and avoids unnecessary I/O. This is known as *incremental checkpointing*.

Note: There are other cases when DBW n may write (for example, when tablespaces are made read-only or are placed offline). In such cases, no incremental checkpoint occurs because dirty buffers belonging only to the corresponding data files are written to the database unrelated to the SCN order.

The LRU algorithm keeps more frequently accessed blocks in the buffer cache to minimize disk reads. A CACHE option can be placed on tables to help retain blocks even longer in memory.

The DB_WRITER_PROCESSES initialization parameter specifies the number of DBW n processes. The maximum number of Database Writer processes is 100. If it is not specified by the user during startup, Oracle Database determines how to set DB_WRITER_PROCESSES based on the number of CPUs and processor groups.

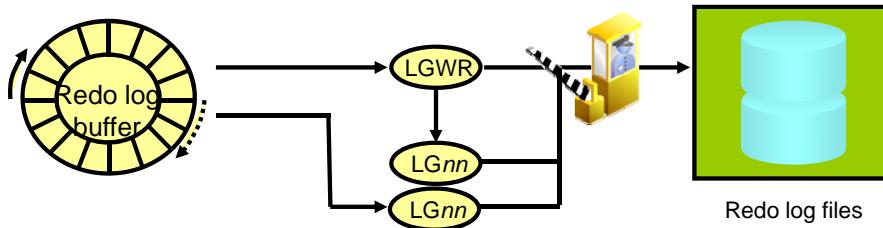
The DBW n process writes dirty buffers to disk under the following conditions:

- When a server process cannot find a clean reusable buffer after scanning a threshold number of buffers, it signals DBW n to write. DBW n writes dirty buffers to disk asynchronously while performing other processing.
- DBW n writes buffers to advance the checkpoint, which is the position in the redo thread (log) from which instance recovery begins. This log position is determined by the oldest dirty buffer in the buffer cache.

In all cases, DBW n performs batched (multiblock) writes to improve efficiency. The number of blocks written in a multiblock write varies by operating system.

Log Writer Process (LGWR)

- Writes the redo log buffer to a redo log file on disk
 - When a user process commits a transaction
 - When an online redo log switch occurs
 - When the redo log buffer is one-third full or contains 1 MB of buffered data
 - Before a DBW n process writes modified buffers to disk
 - When three seconds have passed since the last write
- Serves as coordinator of LG n processes and ensures correct order for operations that must be ordered



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Log Writer process (LGWR) is responsible for redo log buffer management by writing the redo log buffer entries to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote.

LGWR starts and coordinates multiple helper processes that concurrently perform some of the work. LGWR handles the operations that are very fast, or must be coordinated, and delegates operations to the LG n n that could benefit from concurrent operations, primarily writing the redo from the log buffer to the redo log file and posting the completed write to the foreground process that is waiting.

Because LG n n processes work concurrently and certain operations must be performed in order, LGWR forces ordering so that even if the writes complete out of order, the posting to the foreground processes will be in the correct order.

The redo log buffer is a circular buffer. When LGWR writes redo entries from the redo log buffer to a redo log file, server processes can then copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy. LGWR writes one contiguous portion of the buffer to disk.

LGWR writes:

- When a user process commits a transaction
- When an online redo log switch occurs
- When the redo log buffer is one-third full or contains 1 MB of buffered data
- Before a DBW n process writes modified buffers to disk (if necessary)
- When three seconds have passed since the last write to log files

Before DBW n can write a modified buffer, all redo records that are associated with the changes to the buffer must be written to disk (the write-ahead protocol). If DBW n finds that some redo records have not been written, it signals LGWR to write the redo records to disk and waits for LGWR to complete writing the redo log buffer before it can write out the data buffers. LGWR writes to the current log group. If one of the files in the group is damaged or unavailable, LGWR continues writing to other files in the group and logs an error in the LGWR trace file and in the system alert log. If all files in a group are damaged, or if the group is unavailable because it has not been archived, LGWR cannot continue to function.

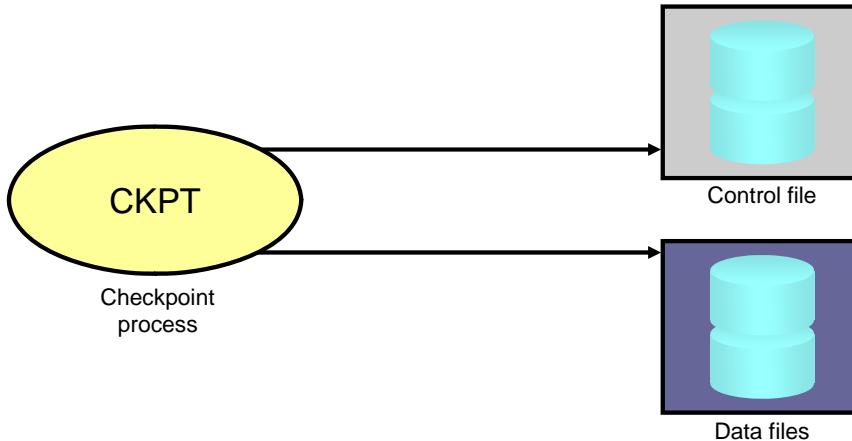
When a user issues a COMMIT statement, LGWR puts a commit record in the redo log buffer and writes it to disk immediately, along with the transaction's redo entries. The corresponding changes to data blocks are deferred until it is more efficient to write them. This is called a *fast commit mechanism*. The atomic write of the redo entry containing the transaction's commit record is the single event that determines whether the transaction has committed. Oracle Database returns a success code to the committing transaction, although the data buffers have not yet been written to disk.

If more buffer space is needed, LGWR sometimes writes redo log entries before a transaction is committed. These entries become permanent only if the transaction is later committed. When a user commits a transaction, the transaction is assigned an SCN, which Oracle Database records along with the transaction's redo entries in the redo log. SCNs are recorded in the redo log so that recovery operations can be synchronized in Real Application Clusters and distributed databases.

In times of high activity, LGWR can write to the redo log file by using group commits. For example, suppose that a user commits a transaction. LGWR must write the transaction's redo entries to disk. As this happens, other users issue COMMIT statements. However, LGWR cannot write to the redo log file to commit these transactions until it has completed its previous write operation. After the first transaction's entries are written to the redo log file, the entire list of redo entries of waiting transactions (not yet committed) can be written to disk in one operation, requiring less I/O than do transaction entries handled individually. Therefore, Oracle Database minimizes disk I/O and maximizes performance of LGWR. If requests to commit continue at a high rate, every write (by LGWR) from the redo log buffer can contain multiple commit records.

Checkpoint Process (CKPT)

- Records checkpoint information in
 - Control file
 - Each data file header
- Signals DBW n to write blocks to disk



ORACLE

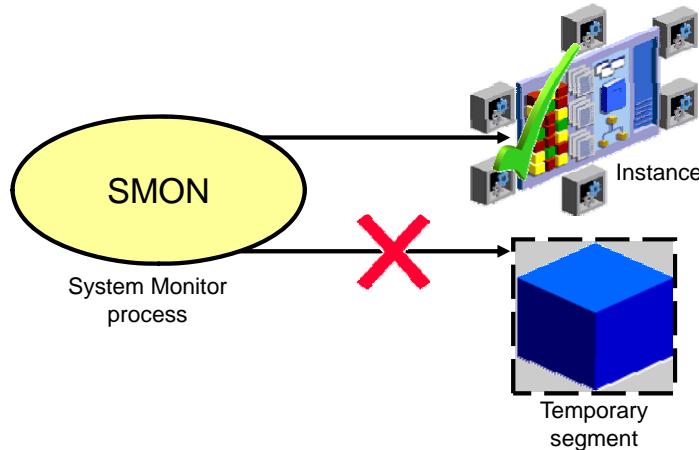
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A *checkpoint* is a data structure that defines a system change number (SCN) in the redo thread of a database. Checkpoints are recorded in the control file and in each data file header. They are a crucial element of recovery.

When a checkpoint occurs, Oracle Database must update the headers of all data files to record the details of the checkpoint. This is done by the CKPT process. The CKPT process does not write blocks to disk; DBW n always performs that work. The SCNs recorded in the file headers guarantee that all changes made to database blocks before that SCN have been written to disk.

System Monitor Process (SMON)

- Performs recovery at instance startup
- Cleans up unused temporary segments



ORACLE

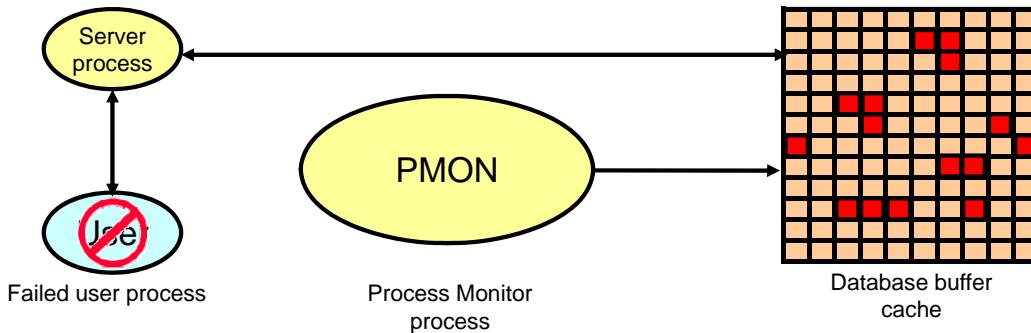
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The System Monitor process (SMON) performs recovery at instance startup if necessary. SMON is also responsible for cleaning up temporary segments that are no longer in use. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online.

SMON checks regularly to see whether the process is needed. Other processes can call SMON if they detect a need for it.

Process Monitor Process (PMON)

- Performs process recovery when a user process fails
 - Cleans up the database buffer cache
 - Frees resources that are used by the user process
- Monitors sessions for idle session timeout



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

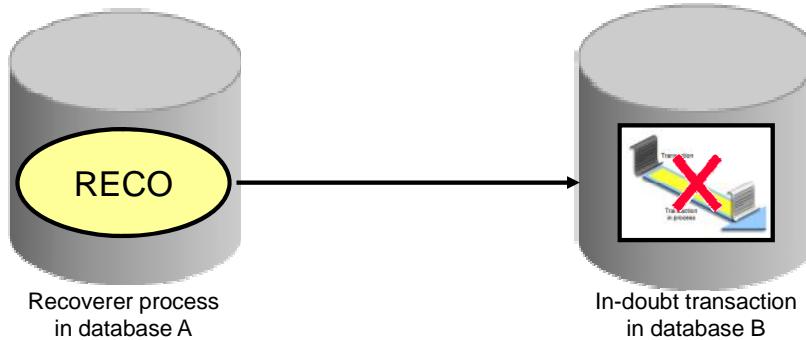
The Process Monitor process (PMON) performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using. For example, it resets the status of the active transaction table, releases locks, and removes the process ID from the list of active processes.

PMON periodically checks the status of dispatcher and server processes, and restarts any that have stopped running (but not any that Oracle Database has terminated intentionally).

Like SMON, PMON checks regularly to see whether it is needed; it can be called if another process detects the need for it.

Recoverer Process (RECO)

- Used with the distributed database configuration
- Automatically connects to other databases involved in in-doubt distributed transactions
- Automatically resolves all in-doubt transactions
- Removes any rows that correspond to in-doubt transactions



ORACLE

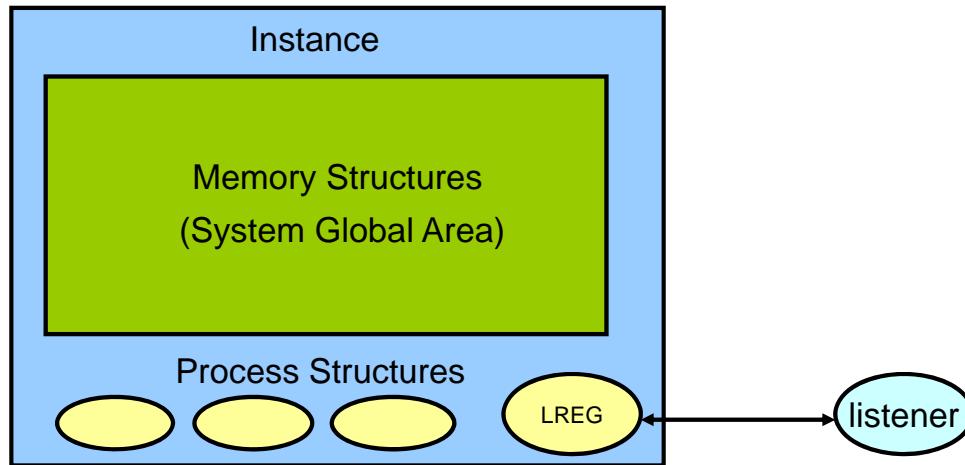
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Recoverer process (RECO) is a background process that is used with the distributed database configuration that automatically resolves failures involving distributed transactions. The RECO process of an instance automatically connects to other databases involved in an in-doubt distributed transaction. When the RECO process re-establishes a connection between involved database servers, it automatically resolves all in-doubt transactions, removing from each database's pending transaction table any rows that correspond to the resolved in-doubt transactions.

If the RECO process fails to connect with a remote server, RECO automatically tries to connect again after a timed interval. However, RECO waits an increasing amount of time (growing exponentially) before it attempts another connection.

Listener Registration Process (LREG)

Registers information about the database instance and dispatcher processes with the Oracle Net Listener



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Listener Registration process, LREG, registers information about the database instance and dispatcher processes with the Oracle Net Listener. LREG provides the listener with the following information:

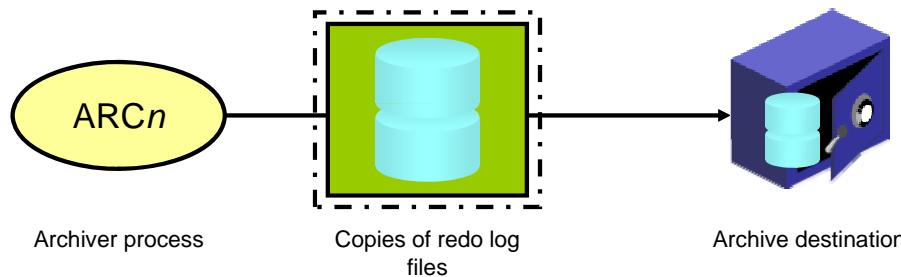
- Names of the database services
- Name of the database instance associated with the services and its current and maximum load
- Service handlers (dispatchers and dedicated servers) available for the instance, including their type, protocol addresses, and current and maximum load

When the instance starts, LREG attempts to connect to the listener. If the listener is running, LREG passes information to it. If the listener is not running, LREG periodically attempts to connect to it. It may take up to 60 seconds for LREG to register the database instance with the listener after the listener has started.

You can use the `ALTER SYSTEM REGISTER` command to immediately initiate service registration after starting the listener.

Archiver Processes (ARCn)

- Copy redo log files to a designated storage device after a log switch has occurred
- Can collect transaction redo data and transmit that data to standby destinations



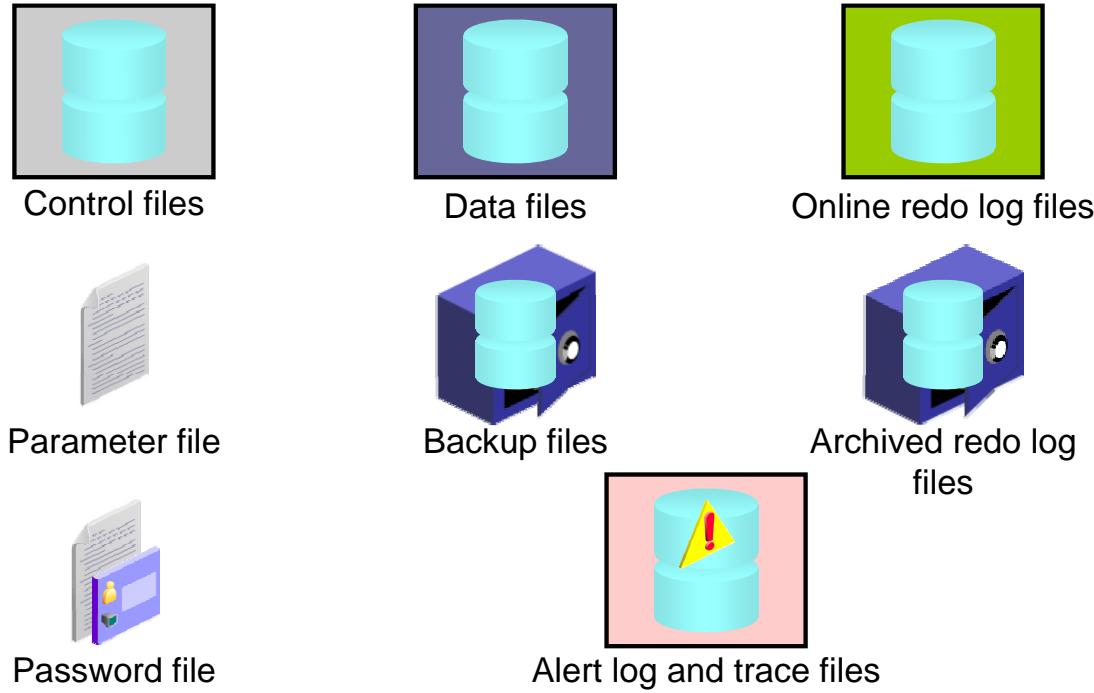
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Archiver processes (ARCn) copy redo log files to a designated storage device after a log switch has occurred. ARCn processes are present only when the database is in ARCHIVELOG mode and automatic archiving is enabled.

If you anticipate a heavy workload for archiving (such as during bulk loading of data), you can increase the maximum number of Archiver processes. There can also be multiple archive log destinations. It is recommended that there be at least one Archiver process for each destination. The default is to have four Archiver processes.

Database Storage Architecture



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The files that comprise an Oracle database are as follows:

- **Control files:** Each database has one unique control file that contains data about the database itself (that is, physical database structure information). Multiple copies may be maintained to protect against total loss. It can also contain metadata related to backups. The control file is critical to the database. Without the control file, the database cannot be opened.
- **Data files:** Contain the user or application data of the database, as well as metadata and the data dictionary
- **Online redo log files:** Allow for instance recovery of the database. If the database server crashes and does not lose any data files, the instance can recover the database with the information in these files.

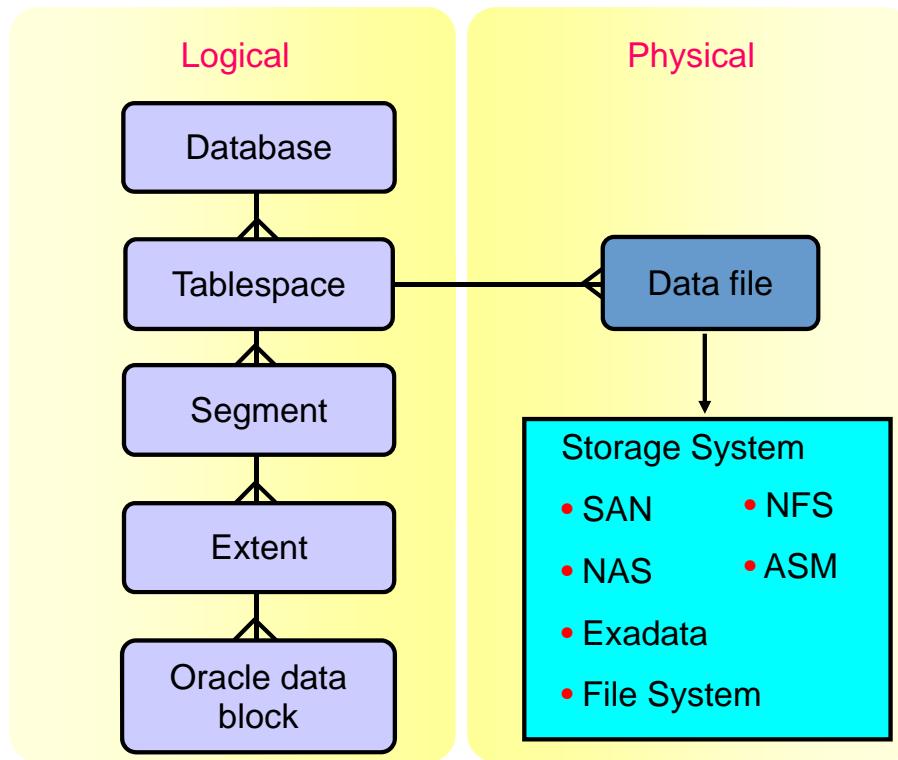
The following additional files are used during the operation of the database:

- **Parameter file:** Is used to define how the instance is configured when it starts up
- **Password file:** Allows users using the SYSDBA, SYSOPER, SYSBACKUP, SYSDG, SYSKM, and SYSASM roles to connect remotely to the instance and perform administrative tasks
- **Backup files:** Are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.

- **Archived redo log files:** Contain an ongoing history of the data changes (redo) that are generated by the instance. Using these files and a backup of the database, you can recover a lost data file. That is, archive logs enable the recovery of restored data files.
- **Trace files:** Each server and background process can write to an associated trace file. When an internal error is detected by a process, the process dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, whereas other information is for Oracle Support Services.
- **Alert log file:** These are special trace entries. The alert log of a database is a chronological log of messages and errors. Oracle recommends that you review the alert log periodically.

Note: Parameter, password, alert, and trace files are covered in other lessons.

Logical and Physical Database Structures



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The database has logical structures and physical structures.

Databases, Tablespaces, and Data Files

The relationship among databases, tablespaces, and data files is illustrated in the slide. Each database is logically divided into two or more tablespaces. One or more data files are explicitly created for each tablespace to physically store the data of all segments in a tablespace. If it is a TEMPORARY tablespace, it has a temporary file instead of a data file. A tablespace's data file can be physically stored on any supported storage technology.

Tablespaces

A database is divided into logical storage units called *tablespaces*, which group related logical structures or data files together. For example, tablespaces commonly group all of an application's segments to simplify some administrative operations.

Data Blocks

At the finest level of granularity, an Oracle database's data is stored in *data blocks*. One data block corresponds to a specific number of bytes of physical space on the disk. A data block size is specified for each tablespace when it is created. A database uses and allocates free database space in Oracle data blocks.

Extents

The next level of logical database space is an *extent*. An extent is a specific number of contiguous Oracle data blocks (obtained in a single allocation) that are used to store a specific type of information. Oracle data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.

Segments

The level of logical database storage above an extent is called a *segment*. A segment is a set of extents allocated for a certain logical structure. Example:

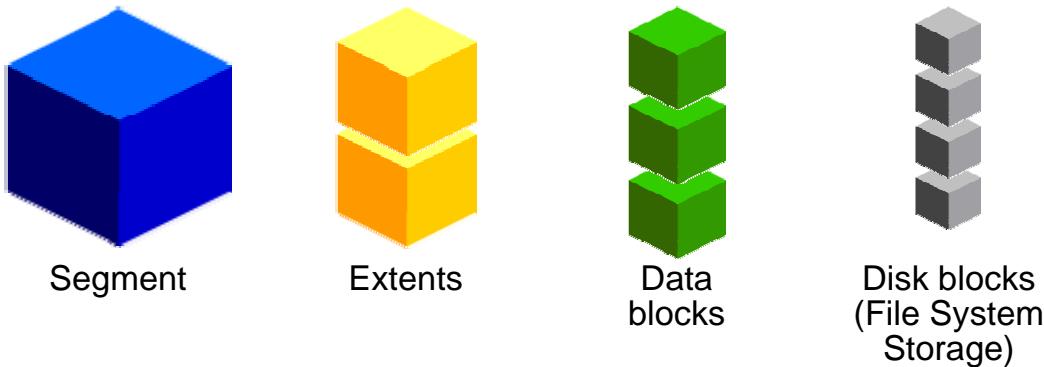
- **Data segments:** Each nonclustered, non-index-organized table has a data segment, with the exception of external tables, global temporary tables, and partitioned tables (in which each table has one or more segments). All of the table's data is stored in the extents of its data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- **Index segments:** Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
- **Undo segments:** One UNDO tablespace is created for each database instance. This tablespace contains numerous undo segments to temporarily store undo information. The information in an undo segment is used to generate read-consistent database information and, during database recovery, to roll back uncommitted transactions for users.
- **Temporary segments:** Temporary segments are created by the Oracle database when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the database for future use. Specify either a default temporary tablespace for every user, or a default temporary tablespace that is used database-wide.

Note: There are other types of segments not listed here. There are also schema objects such as views, packages, triggers, and so on that are not considered segments even though they are database objects. A segment owns its respective disk space allocation. The other objects exist as rows stored in a system metadata segment.

The Oracle Database server dynamically allocates space. When the existing extents of a segment are full, additional extents are added. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on the disk, and they can come from different data files belonging to the same tablespace.

Segments, Extents, and Blocks

- Segments exist in a tablespace.
- Segments are collections of extents.
- Extents are collections of data blocks.
- Data blocks are mapped to disk blocks.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A subset of database objects such as tables and indexes are stored as segments in tablespaces. Each segment contains one or more extents. An extent consists of contiguous data blocks, which means that each extent can exist only in one data file. Data blocks are the smallest unit of I/O in the database.

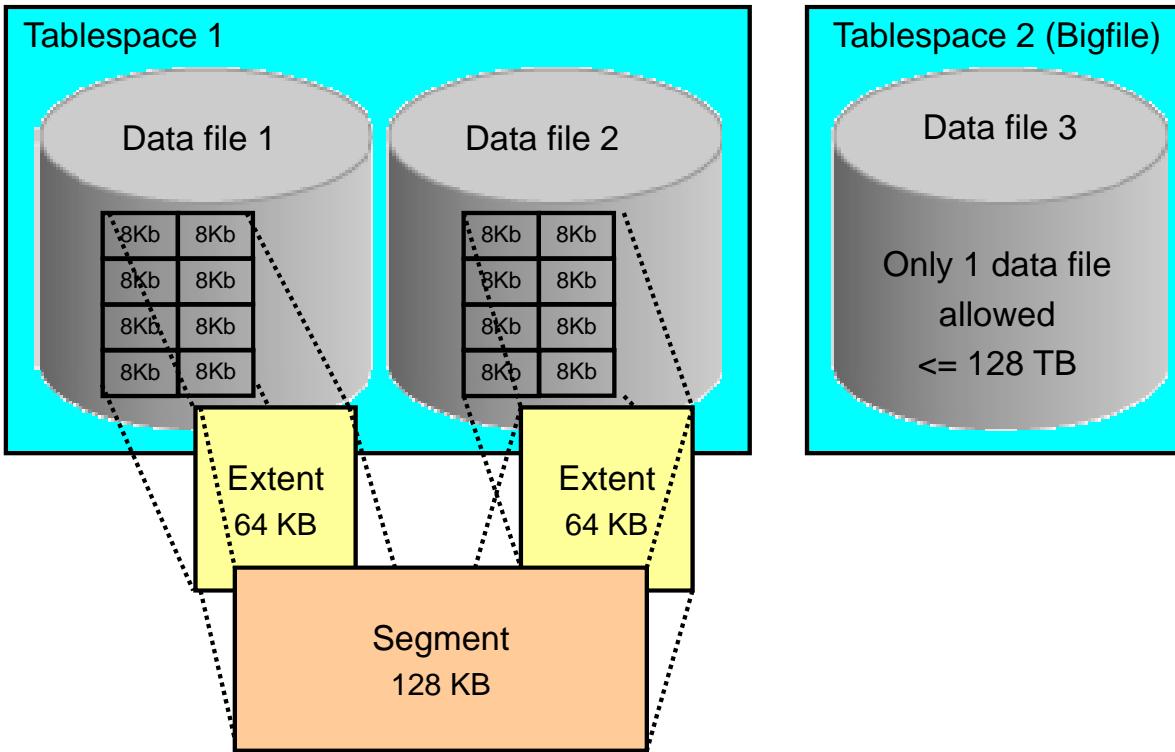
When the database requests a set of data blocks from the operating system (OS), the OS maps this to an actual file system or disk block on the storage device. Because of this, you do not need to know the physical address of any of the data in your database. This also means that a data file can be striped or mirrored on several disks.

The size of the data block can be set at the time of database creation. The default size of 8 KB is adequate for most databases. If your database supports a data warehouse application that has large tables and indexes, a larger block size may be beneficial.

If your database supports a transactional application in which reads and writes are random, specifying a smaller block size may be beneficial. The maximum block size depends on your OS. The minimum Oracle block size is 2 KB; it should rarely (if ever) be used.

You can have tablespaces with a nonstandard block size. For details, see the *Oracle Database Administrator's Guide*.

Tablespaces and Data Files



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A database is divided into *tablespaces*, which are logical storage units that can be used to group related logical structures. One or more data files are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace.

The graphic in the slide illustrates tablespace 1, composed of two data files. A segment of 128 KB size, composed of two extents is spanning the two data files. The first extent of size 64 KB is in the first data file and the second extent, also of size 64 KB is in the second data file. Both extents are formed from contiguous 8 KB Oracle blocks.

Note: You can also create bigfile tablespaces, which have only one file that is often very large. The file may be any size up to the maximum that the row ID architecture permits. The maximum size is the block size for the tablespace multiplied by 2^{36} , or 128 TB for a 32 KB block size. Traditional smallfile tablespaces (which are the default) may contain multiple data files, but the files cannot be as large. For more information about bigfile tablespaces, see the *Oracle Database Administrator's Guide*.

SYSTEM and SYSAUX Tablespaces

- The SYSTEM and SYSAUX tablespaces are mandatory tablespaces that are created at the time of database creation. They must be online.
- The SYSTEM tablespace is used for core functionality (for example, data dictionary tables).
- The auxiliary SYSAUX tablespace is used for additional database components.
- The SYSTEM and SYSAUX tablespaces should not be used for application data.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each Oracle database must contain a SYSTEM tablespace and a SYSAUX tablespace. They are automatically created when the database is created. The system default is to create a smallfile tablespace. You can also create bigfile tablespaces, which enable the Oracle database to manage ultralarge files.

A tablespace can be online (accessible) or offline (not accessible). The SYSTEM tablespace is always online when the database is open. It stores tables that support the core functionality of the database, such as the data dictionary tables.

The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace. The SYSAUX tablespace stores many database components, and it must be online for the correct functioning of all database components. The SYSTEM and SYSAUX tablespaces are not recommended for storing an application's data. Additional tablespaces can be created for this purpose.

Note: The SYSAUX tablespace may be taken offline to perform tablespace recovery, whereas this is not possible for the SYSTEM tablespace. Neither of them may be made read-only.

Oracle Container Database: Introduction

- *Pluggable database*: Is a set of database schemas that appears logically to users and applications as a separate database
- *Multitenant container database*: Has a database instance and database files at the physical level
- All pluggable databases share:
 - Background processes
 - Shared/process memory
 - Oracle metadata



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A pluggable database (PDB) 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 (CDB) has a database instance and database files, just as a non-container database does.

It is easy to plug non-CDBs into a CDB.

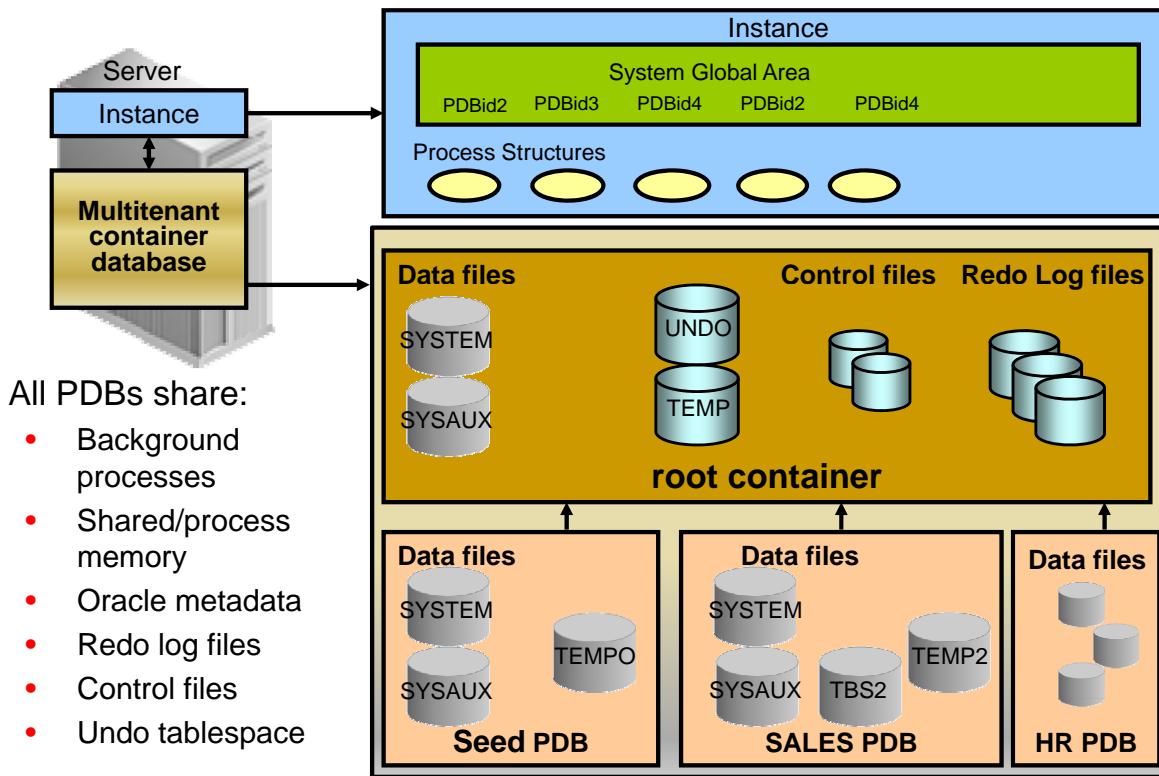
A CDB avoids redundancy of:

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

A CDB grouping several applications has 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.

Multitenant Architecture



ORACLE

Copyright © 2014, 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 (HR and SALES) use a single instance and are maintained separately.

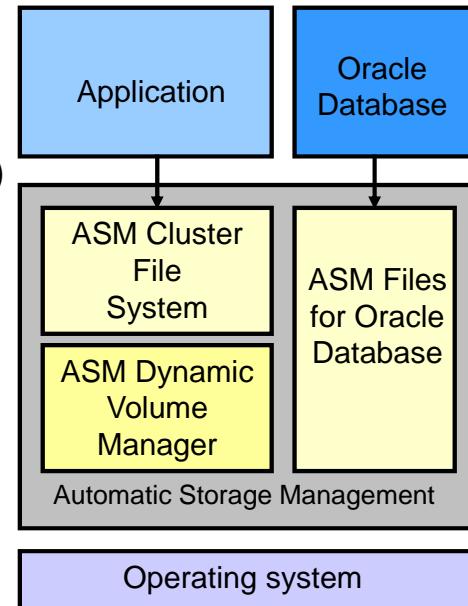
At the physical level, the CDB has a database instance and database files, just as a non-CDB does.

- 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*.

Refer to the *Oracle Database 12c: Managing Multitenant* course for detailed information.

Automatic Storage Management

- Is a portable and high-performance cluster file system
- Manages Oracle database files
- Manages application files with ASM Cluster File System (ACFS)
- Spreads data across disks to balance load
- Mirrors data in case of failures
- Solves storage management challenges



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Automatic Storage Management (ASM) provides vertical integration of the file system and the volume manager for Oracle database files. ASM can provide management for single symmetric multiprocessing (SMP) machines or across multiple nodes of a cluster for Oracle Real Application Clusters (RAC) support.

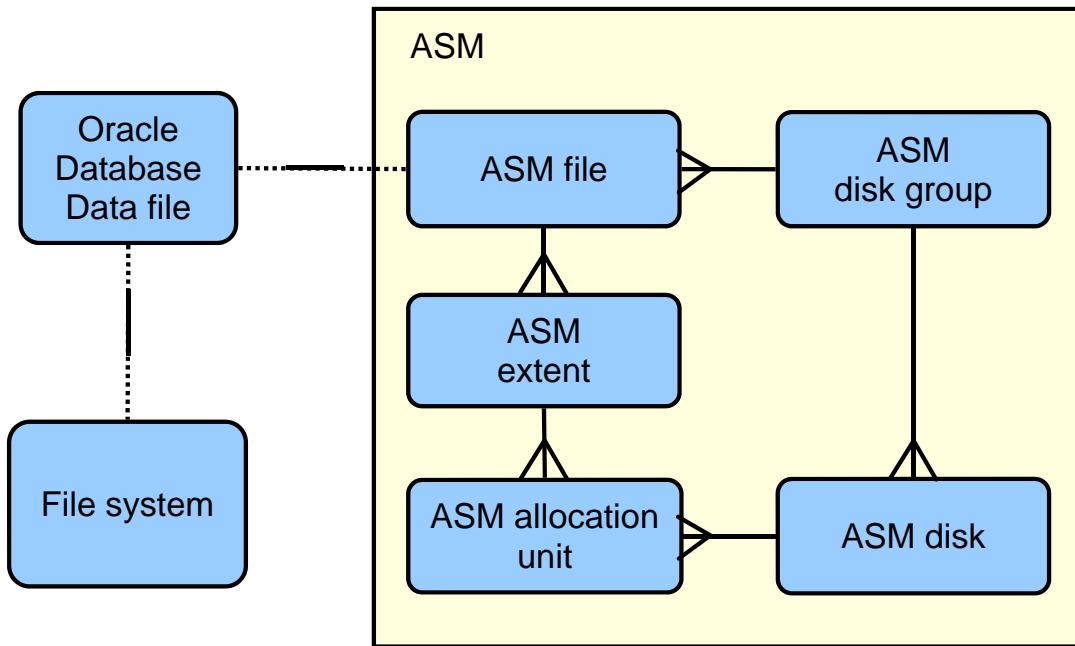
Oracle ASM Cluster File System (ACFS) is a multi-platform, scalable file system, and storage management technology that extends ASM functionality to support application files outside of the Oracle Database such as executables, reports, BFILEs, video, audio, text, images, and other general-purpose application file data.

ASM distributes input/output (I/O) load across all available resources to optimize performance while removing the need for manual I/O tuning. ASM helps database administrators (DBAs) manage a dynamic database environment by enabling them to increase the database size without having to shut down the database to adjust storage allocation.

ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied storage mechanisms. Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per-file basis.

ASM capabilities save the DBA's time by automating manual storage and thereby increasing the administrator's ability to manage more and larger databases with increased efficiency.

ASM Storage Components



ORACLE

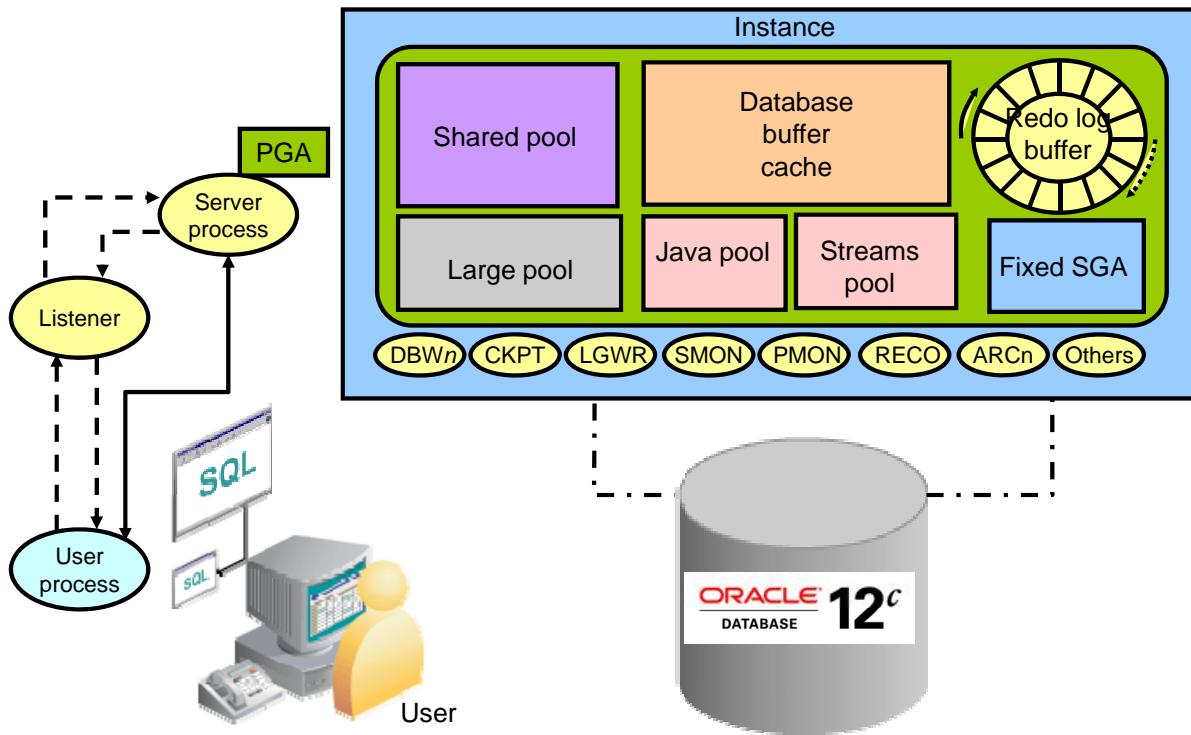
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ASM does not eliminate any existing database functionality. Existing databases are able to operate as they always have. New files may be created as ASM files, whereas existing ones are administered in the old way or can be migrated to ASM.

The diagram illustrates the relationships between an Oracle database data file and the ASM storage components. The crow's foot notation represents a one-to-many relationship. An Oracle Database data file has a one-to-one relationship with either a file stored on the operating system in a file system or an ASM file.

An Oracle ASM disk group is a collection of one or more Oracle ASM disks managed as a logical unit. The data structures in a disk group are self-contained using some of the space for metadata needs. Oracle ASM disks are the storage devices provisioned to an Oracle ASM disk group and can be physical disk or partitions, a Logical Unit Number (LUN) from a storage array, a logical volume (LV), or a network-attached file. Each ASM disk is divided into many ASM allocation units, the smallest contiguous amount of disk space that ASM allocates. When you create an ASM disk group, you can set the ASM allocation unit size to 1, 2, 4, 8, 16, 32, or 64 MB depending on the disk group compatibility level. One or more ASM allocation units forms an ASM extent. An Oracle ASM extent is the raw storage used to hold the contents of an Oracle ASM file. An Oracle ASM file consists of one or more file extents. Variable extent sizes of 1*AU size, 4*AU size, and 16*AU size are used for supporting very large ASM files.

Interacting with an Oracle Database: Memory, Processes, and Storage



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following example describes Oracle database operations at the most basic level. It illustrates an Oracle database configuration in which the user and associated server process are on separate computers, connected through a network.

1. An instance has started on a node where Oracle Database is installed, often called the *host* or *database server*.
2. A user starts an application spawning a user process. The application attempts to establish a connection to the server. (The connection may be local, client/server, or a three-tier connection from a middle tier.)
3. The server runs a listener that has the appropriate Oracle Net Services handler. The listener detects the connection request from the application and creates a dedicated server process on behalf of the user process.
4. The user runs a DML-type SQL statement and commits the transaction. For example, the user changes the address of a customer in a table and commits the change.
5. The server process receives the statement and checks the shared pool (an SGA component) for any shared SQL area that contains an identical SQL statement. If a shared SQL area is found, the server process checks the user's access privileges to the requested data, and the existing shared SQL area is used to process the statement. If a shared SQL area is not found, a new shared SQL area is allocated for the statement so that it can be parsed and processed.

6. The server process retrieves any necessary data values, either from the actual data file (table) or from values stored in the database buffer cache.
7. The server process modifies data in the SGA. Because the transaction is committed, the Log Writer process (LGWR) immediately records the transaction in the redo log file. The Database Writer process (DBW n) writes modified blocks permanently to disk when it is efficient to do so.
8. If the transaction is successful, the server process sends a message across the network to the application. If it is not successful, an error message is transmitted.
9. Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

Quiz

The Process Monitor process (PMON):

- a. Performs recovery at instance startup
- b. Performs process recovery when a user process fails
- c. Automatically resolves all in-doubt transactions
- d. Writes the redo log buffer to a redo log file



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- List the major architectural components of Oracle Database
- Explain memory structures
- Describe background processes
- Correlate logical and physical storage structures
- Describe pluggable databases
- Describe ASM storage components



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This is a paper practice with questions about:

- Database architecture
- Memory
- Processes
- File structures



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Oracle University and Error : You are not a Valid Partner use only

Oracle Database Management Tools



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use SQL*Plus to access the Oracle database
- Use Oracle Enterprise Manager Database Express to perform administrative tasks
- Use Oracle Enterprise Manager Cloud Control to manage the database instance



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Database Management Tools: Introduction

- SQL*Plus provides an interface to your database so that you can:
 - Perform database management operations
 - Execute SQL commands to query, insert, update, and delete data in your database
- SQL Developer
 - Is a graphical user interface for accessing your instance of Oracle Database
 - Supports development in both SQL and PL/SQL
 - Is available in the default installation of Oracle Database
- Oracle Enterprise Manager Database Express
- Oracle Enterprise Manager Cloud Control



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

SQL*Plus is a command-line program that you use to submit SQL and PL/SQL statements to an Oracle database. You can submit statements interactively or as SQL*Plus scripts.

SQL*Plus is installed with the database and is located in your \$ORACLE_HOME/bin directory.

You can start SQL*Plus from the command line, or from the Start menu on a Windows client.

SQL Developer is a graphical user interface for accessing your instance of Oracle Database. SQL Developer supports development in both the SQL and PL/SQL languages. It is available in the default installation of Oracle Database.

With SQL Developer, you can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. You can also run any number of provided reports, as well as create and save your own.

Using SQL*Plus

SQL*Plus is:

- A command-line tool
- Used interactively or in batch mode

```
$ sqlplus hr

SQL*Plus: Release 12.1.0.2.0 Production on Mon Oct 6 13:28:12 2014

Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter password:
Last Successful login time: Mon Oct 06 2014 13:24:35 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> SELECT last_name FROM employees;
LAST_NAME
-----
Abel
Ande
...
...
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can use the SQL*Plus command-line interface to execute SQL*Plus, SQL, and PL/SQL commands to:

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

To start SQL*Plus:

1. Open a terminal window.
2. At the command-line prompt, enter the SQL*Plus command in the following form:
 \$ sqlplus <userid>/<pwd> or /nolog
3. If you use the NOLOG option, you must enter CONNECT followed by the username you want to connect as.
 SQL> connect <username>
4. When prompted, enter the user's password. SQL*Plus starts and connects to the default database.

Calling SQL*Plus from a Shell Script

```
$ ./batch_sqlplus.sh
SQL*Plus: Release 12.1.0.1.0 Production on Thu Nov 15 09:10:48 2012
Copyright (c) 1982, 2012, Oracle. All rights reserved.

Last Successful login time: Wed Nov 14 2012 12:10:11 +00:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real Application Testing and
Unified Auditing options

SQL>
  COUNT(*)
-----
     107
SQL>
107 rows updated.
SQL>
Commit complete.
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.1.0
- 64bit Production
With the Partitioning, OLAP, Advanced Analytics, Real Application Testing
and Unified Auditing options
$
```

Output



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can call SQL*Plus from a shell script or BAT file by invoking `sqlplus` and using the operating system scripting syntax for passing parameters.

In this example, the SELECT, UPDATE, and COMMIT statements are executed before SQL*Plus returns control to the operating system.

Calling a SQL Script from SQL*Plus

script.sql

```
select * from departments where location_id = 1400;  
quit
```

Output

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

```
SQL*Plus: Release 12.1.0.1.0 Production on Thu Nov 15 09:32:36 2012
```

```
Copyright (c) 1982, 2012, Oracle. All rights reserved.
```

```
Last Successful login time: Thu Nov 15 2012 09:30:49 +00:00
```

```
Connected to:
```

```
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production  
With the Partitioning, OLAP, Advanced Analytics, Real Application Testing  
and Unified Auditing options
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
60	IT	103	1400

```
Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit  
Production
```

```
With the Partitioning, OLAP, Advanced Analytics, Real Application Testing  
and Unified Auditing options
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can call an existing SQL script file from within SQL*Plus. This can be done at the command line when first invoking SQL*Plus, as shown in the slide. It can also be done from inside a SQL*Plus session simply by using the "@" operator. For example, this example shows executing the script from within an already established SQL*Plus session:

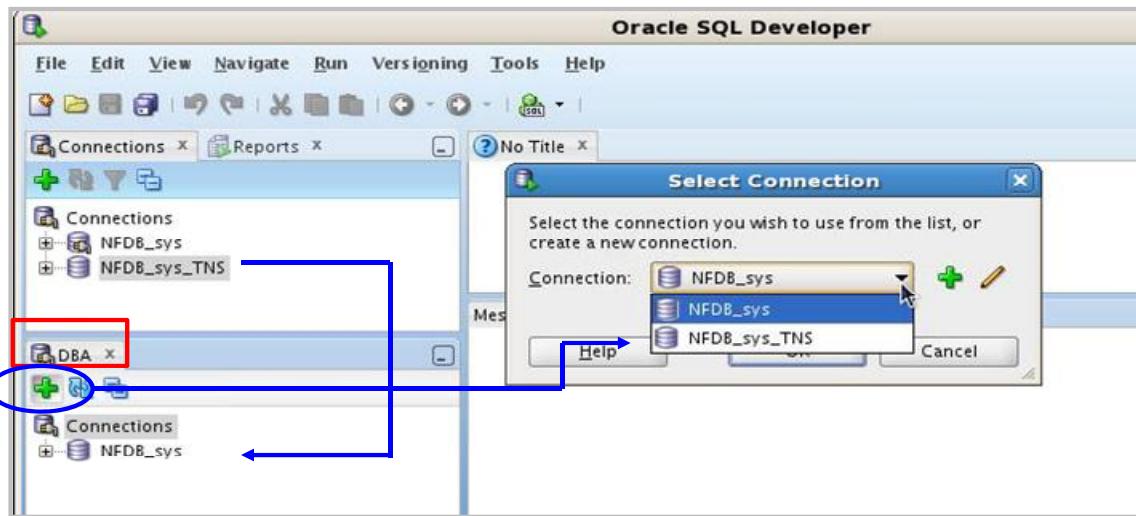
```
SQL> @script.sql
```

Note: The default file extension for script files is .sql. When a script is saved from SQL*Plus by using the SAVE command, this extension is automatically supplied. Scripts with this extension can be executed without supplying the extension at execution time, as in the following example:

```
SQL> @script
```

Oracle SQL Developer: Connections

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



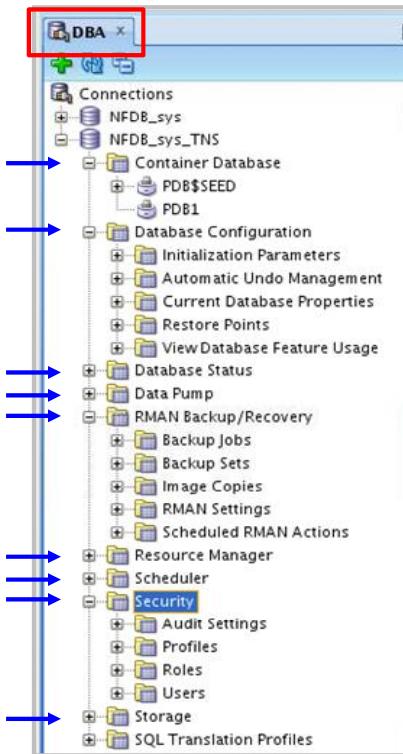
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

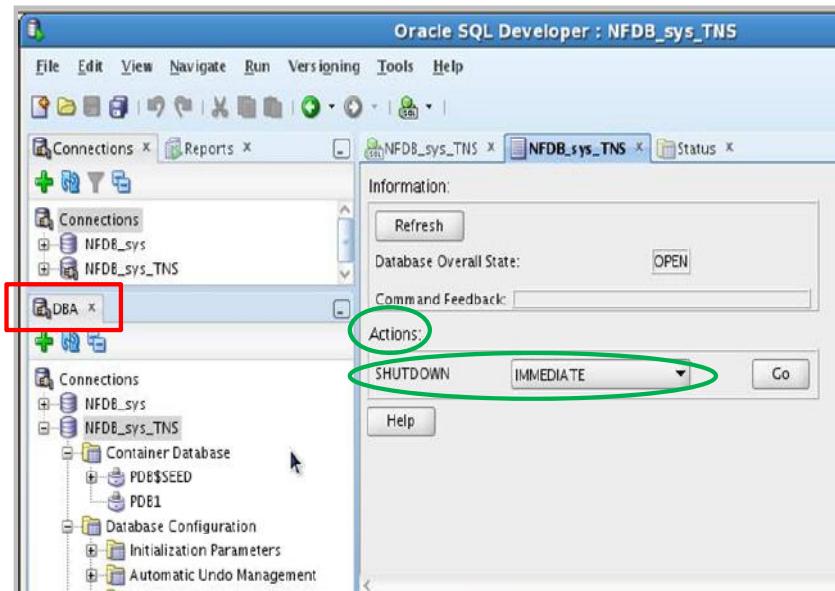
Oracle SQL Developer is a tool that allows stand-alone graphical browsing and development of database schema objects, as well as execution of database administrative tasks. SQL Developer enables users with database administrator privileges to view and edit certain information relevant to DBAs and perform 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

Using DBA features through DBA navigator



Performing DBA actions



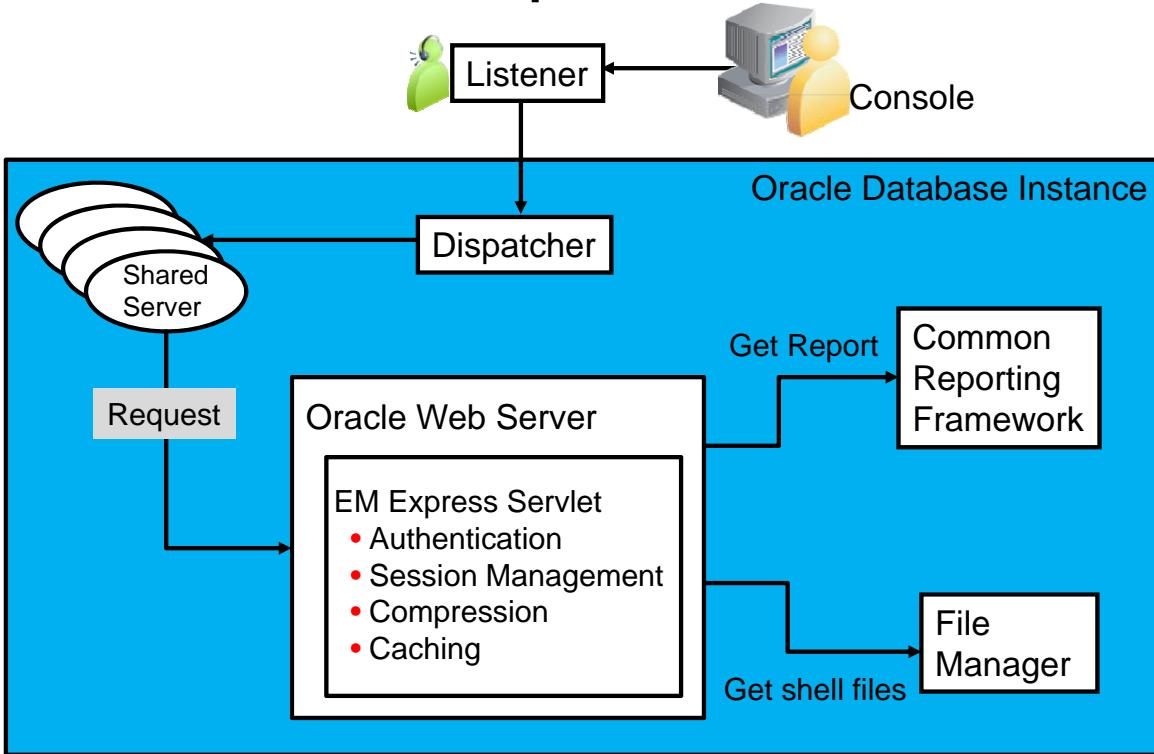
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The DBA operations that can be performed are the following:

- 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, and users
- Storage configuration for archive logs, control files, data files, redo log groups, tablespaces, and temporary tablespace groups

Oracle Enterprise Manager Database Express Architecture



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Enterprise Manager Database Express is a lightweight administration tool. It provides an out-of-box browser-based management solution for a single Oracle database (or database cluster), including performance monitoring, configuration management, administration, diagnostics and tuning.

Oracle Enterprise Manager Database 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 Enterprise Manager Database Express servlet handles the requests, including authentication, session management, compression, and caching. The servlet passes requests for reports to the Common Reporting Framework and actions requiring shell files to the File Manager.

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.

Configuring Enterprise Manager Database Express

- Configure an HTTP listener port for each database instance.
 - Verify DISPATCHERS parameter.

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

- Use DBMS_XDB.setHTTPPort procedure.

```
exec DBMS_XDB.setHTTPPort(5500)
```

- Launch Enterprise Manager Database Express:

```
http://hostname:5500/em
```

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle 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 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.

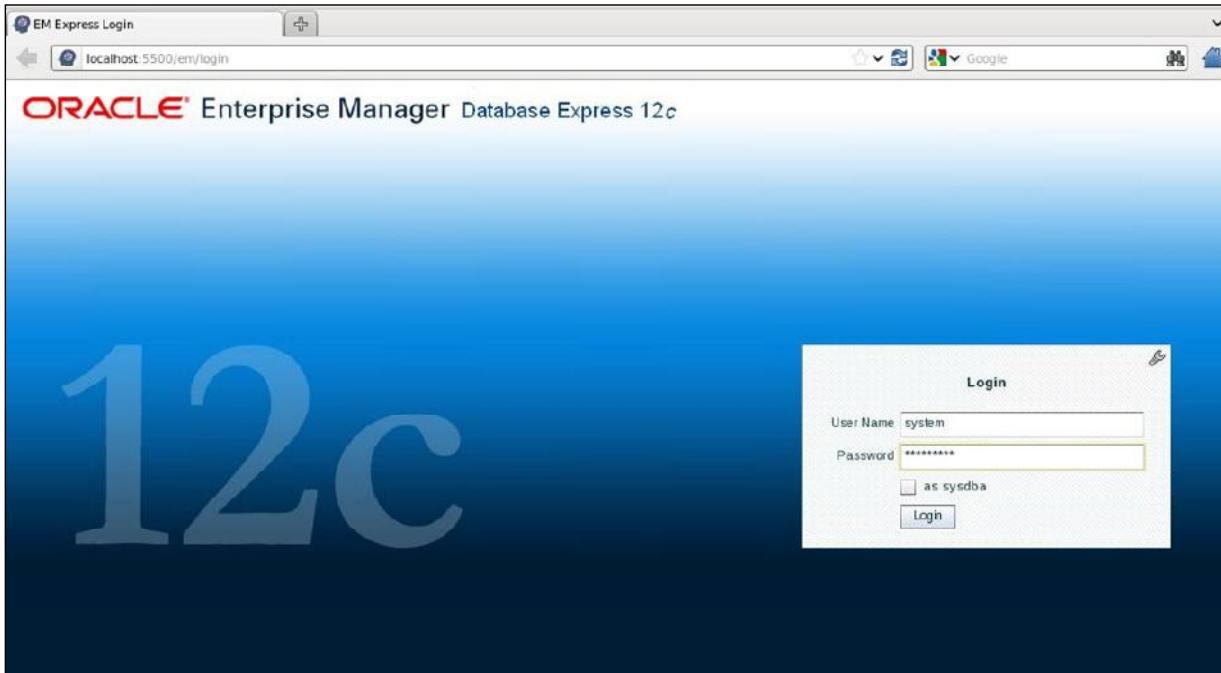
Use the SETHTTPPORT procedure in the DBMS_XDB package to configure a port on the server. Connect to the Enterprise Manager Database Express console with the URL shown in the slide. Substitute the host name of the server and the port number you set by using the SETHTTPPORT procedure.

If you have multiple database instances to monitor on the same machine, set a different port for each. To find the port used for each database instance, execute the following statement:

```
SQL> SELECT dbms_xdb.gethttpport FROM DUAL;
```

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

Logging In to Oracle Enterprise Manager Database Express

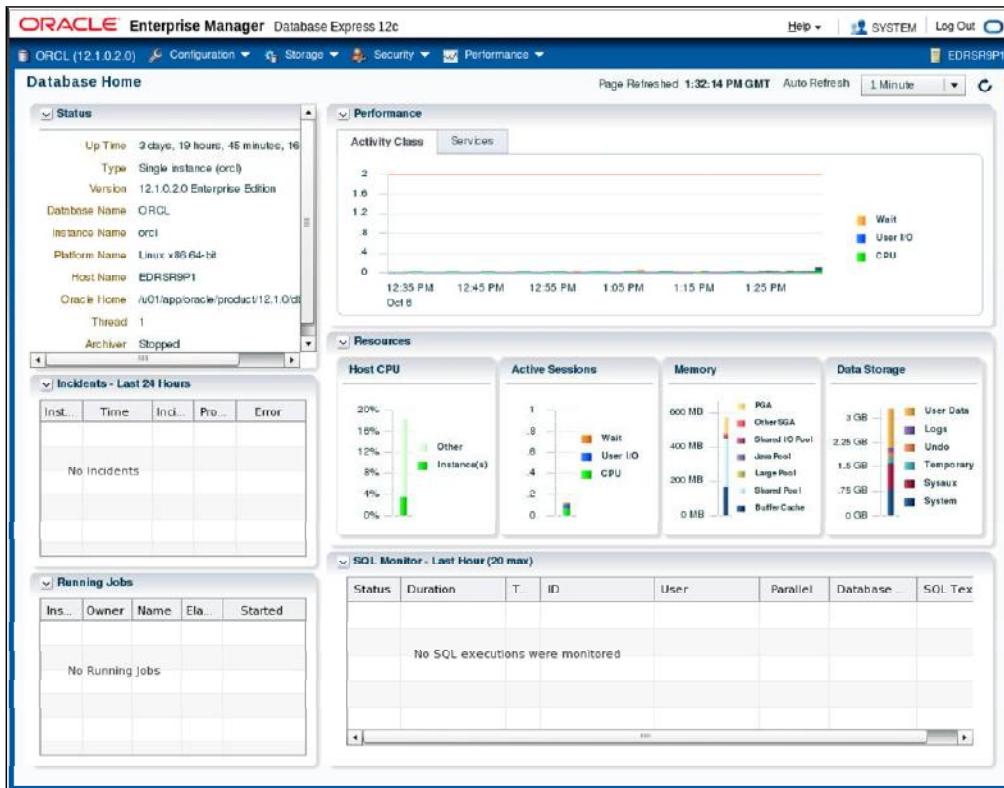


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Launch Enterprise Manager Database Express by using the configured HTTP port. Log in as the database user appropriate to the tasks you want to accomplish.

Using the Database Home Page

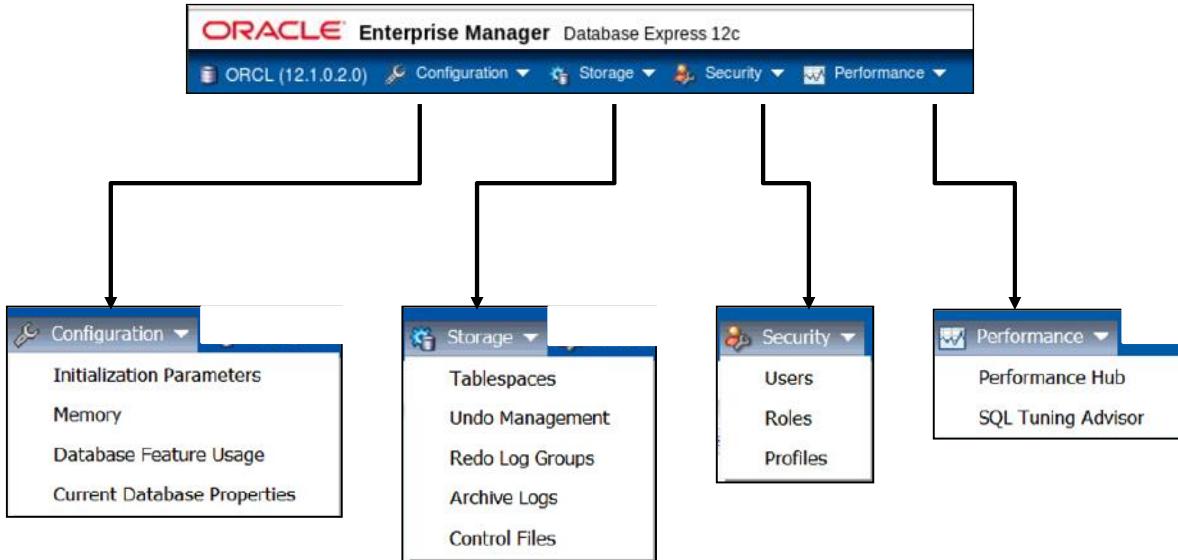


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Using Enterprise Manager Database Express Menus



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

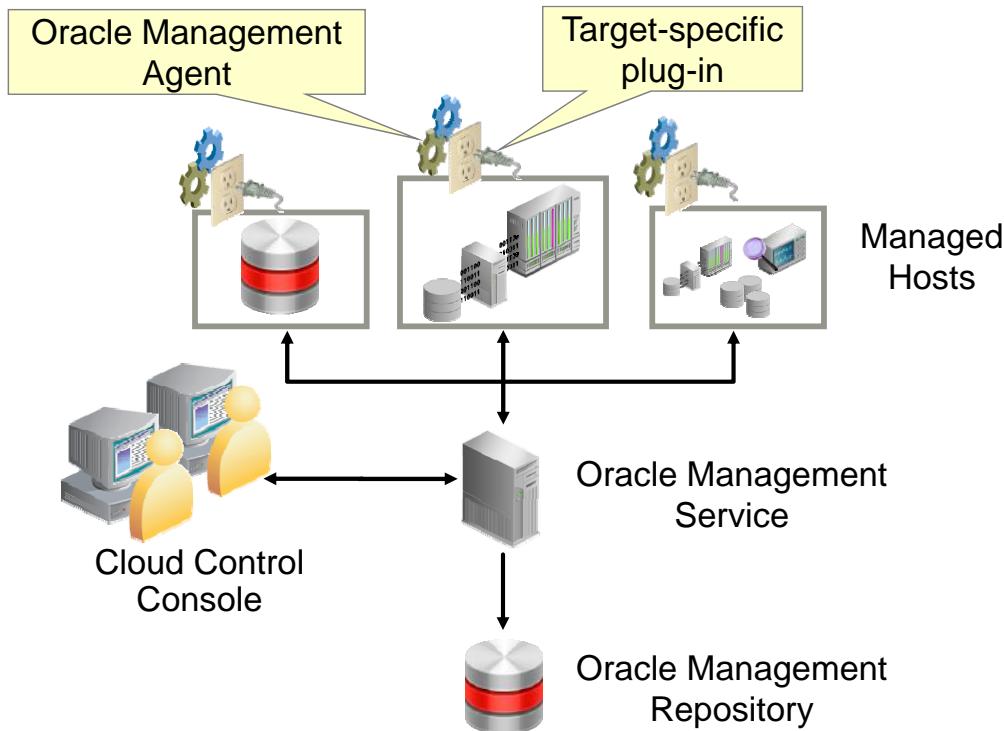


The menu layout for Enterprise Manager Database Express is shown in the slide. There are four main menu items: Configuration, Storage, Security, and Performance. The menu selections for each of the main items are shown.

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 will display a page that allows you to search, view, and modify current and server parameter file (SPFILE) initialization parameters. In many of the pages, when you select an action, a pop-up dialog box enables you to specify parameters. A SQL command is then created to perform the action. You may view the SQL command before you submit it, or you can copy and paste the SQL command.

Oracle Enterprise Manager Cloud Control Components



ORACLE

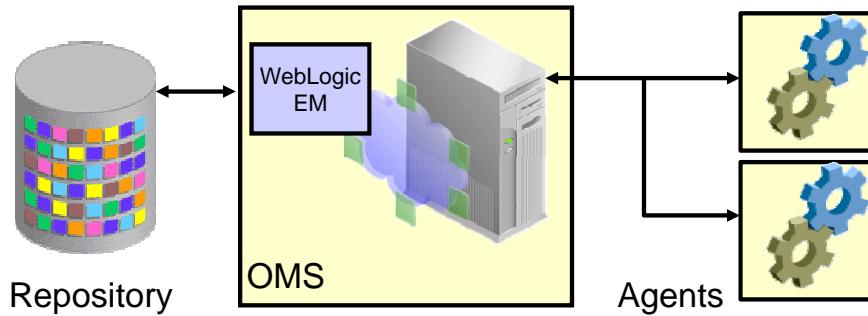
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

Controlling the Enterprise Manager Cloud Control Framework



Component Control Utilities		
Repository	OMS	Agent
SQL*Plus or Server Control	Enterprise Manager Control	Enterprise Manager Control
Listener Control		



Copyright © 2014, 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 Server Control commands; for single instances, there is a choice between SQL*Plus and Server Control. Server Control can be used when Oracle Restart is installed and the database is registered with the OLR.

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

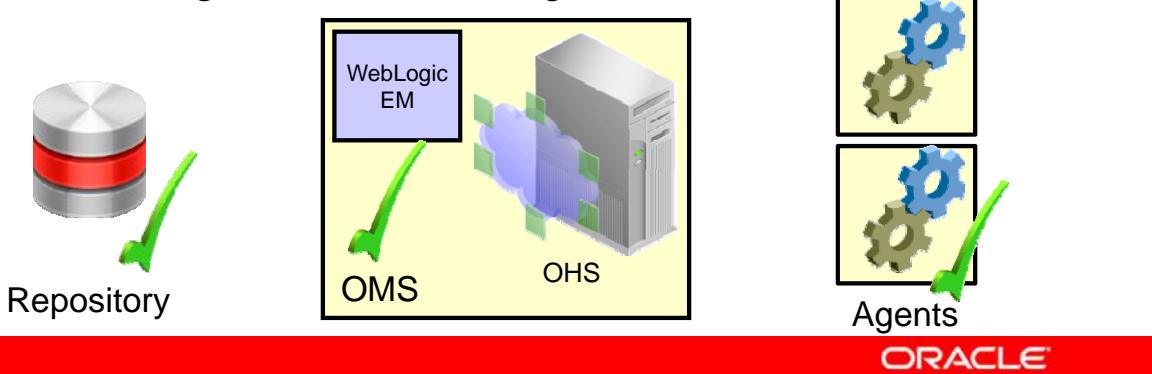
Examples:

```
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 OMS.
4. Start the agent on the OMS/repository server.
5. Start the agents on the managed servers.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

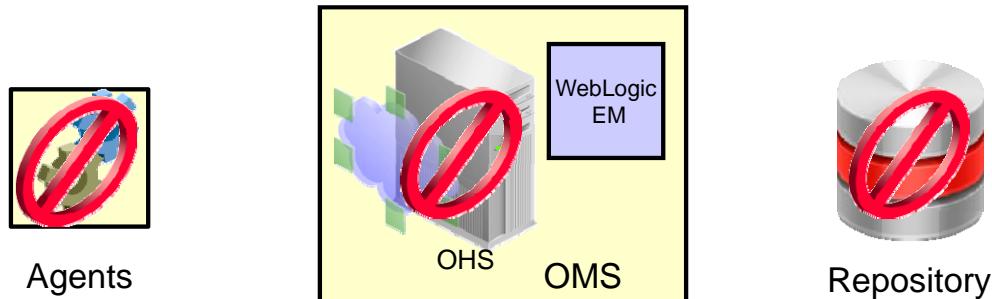
1. Start the repository listener:
`$ORACLE_HOME/bin/lsnrctl start`
2. Start the repository database instance:
`$ORACLE_HOME/bin/sqlplus / as sysdba
SQL> startup`
3. Start OMS (including OHS and WebLogic Managed Server):
`$OMS_HOME/bin/emctl start oms`
4. Start the agent (on OMS/repository host):
`$AGENT_HOME/bin/emctl 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 or the repository is controlled by Oracle Restart.

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 OMS.
4. Stop the repository database instance.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

Types of Enterprise Manager Cloud Control Targets

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 © 2014, 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.

Enterprise Manager Cloud Control

The screenshot shows the Oracle Enterprise Manager Cloud Control interface. The main navigation bar at the top includes links for Enterprise, Targets, Favorites, History, Setup, Help, Guest SUPER ADMIN, and Log Out. The date and time displayed are Page Refreshed Aug 22, 2011 2:29:31 PM UTC.

Enterprise Summary

- Overview:** Targets Monitored: 1420. Status: Targets with Status: 1006, Targets with Pending Activation: 122. A pie chart shows the distribution of target status: Down(133) 13%, Metric Collection 2%, Agent Unreachable(138) 14%, Pending(44) 4%, Up(677) 67%.
- Incidents:** Open: 816, Updated in last 24 hours: 333. A table shows incident categories: Availability (238), Performance (-), Security (-), Others (533), with counts 13, 2, 18, 23 respectively.
- Problems:** Open: 57, Without Service Request: 57, Updated in last 24 hours: 33. A table shows problem types: Suspended Executions (last 7 days) 27, Problem Executions (last 7 days) 54, Action Required Executions (last 7 days) 0.
- Patch Recommendations:** View by Classification or Target Type. Other Recommendations: Security (50).
- Inventory and Usage:** Shows hosts and OS patches. Platform: Enterprise Linux AS release 4 (October Update 8), Enterprise Linux Server release 5.5 (Carthage), Enterprise Linux Server release 5.4 (Carthage), SunOS. Hosts | OS Patches: 23 No, 19 No, 9 No, 2 No, 1 No.
- Compliance Summary:** Compliance Frameworks and Compliance Standards. No data is displayed.
- Least Compliant Targets:** A table showing targets and their compliance scores. Target Name: oracle.com, Host. Standard Evaluations: 1, 0, 0, 4, 0, 0. Violations: 0, 0, 0, 0, 0, 0. Average Compliance Score (%): 51.
- Service Requests:** Sign in to My Oracle Support. Enter your Single Sign-On username and password. User Name: [redacted], Password: [redacted]. Go button. Lost your password?

ORACLE

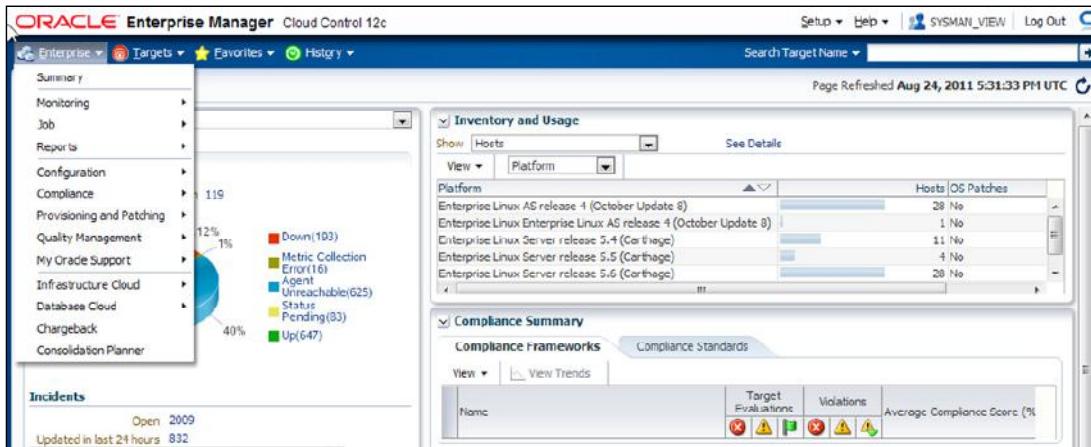
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The screenshot 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)

Using Enterprise Manager Cloud Control

- Predefined home page based on roles
- Setting any page as home page
- Menu-based navigation
- Making any page “favorite” for quick access



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 to be your home page instead.
- Mark any page as a “favorite” for quick access. For example, if there are certain targets that you manage quite often, you can mark the page you manage them from as a favorite in 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.

Quiz

Enterprise Manager Database Express can be used to manage many databases concurrently.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Use SQL*Plus to access the Oracle database
- Use Oracle Enterprise Manager Database Express to perform administrative tasks
- Use Oracle Enterprise Manager Cloud Control to manage the database instance



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Connecting to the database by using SQL*Plus
- Navigating in Enterprise Manager Database Express
- Navigating in Enterprise Manager Cloud Control



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Oracle University and Error : You are not a Valid Partner use only

Managing the Database Instance

4

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

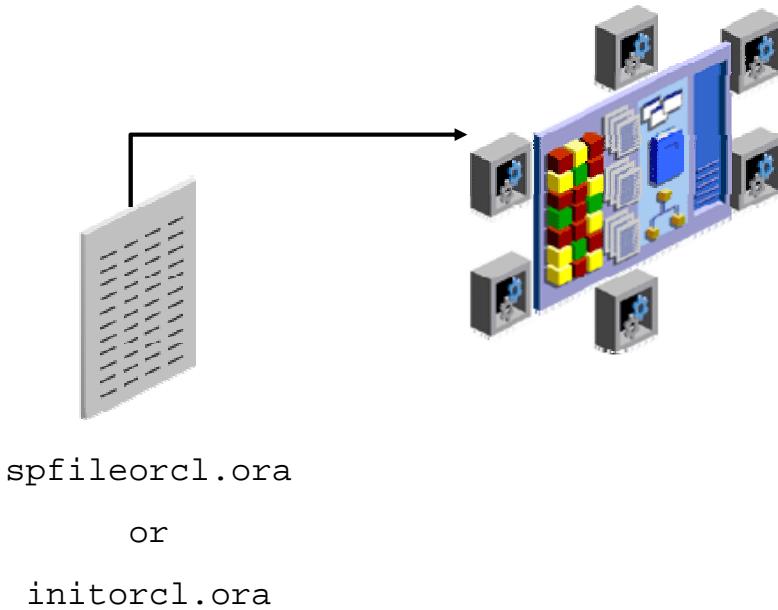
After completing this lesson, you should be able to:

- Start and stop the Oracle database instance and components
- Modify database initialization parameters
- Describe the stages of database startup
- Describe database shutdown options
- View the alert log
- Access dynamic performance views



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Initialization Parameter Files



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When you start the instance, an initialization parameter file is read. There are two types of parameter files.

- **Server parameter file (SPFILE):** This is the preferred type of initialization parameter file. It is a binary file that can be written to and read by the database server and *must not be edited manually*. It resides on the server on which the Oracle instance is executing; it is persistent across shutdown and startup. The default name of this file, which is automatically sought at startup, is `spfile<SID>.ora`.
- **Text initialization parameter file:** This type of initialization parameter file can be read by the database server, but it is not written to by the server. The initialization parameter settings must be set and changed manually by using a text editor so that they are persistent across shutdown and startup. The default name of this file (which is automatically sought at startup if an SPFILE is not found) is `init<SID>.ora`.

It is recommended that you create an SPFILE as a dynamic way to maintain initialization parameters.

Note: The Oracle Database server searches the `$ORACLE_HOME/dbs` directory on Linux for the initialization files.

Types of Values for Initialization Parameters

The Oracle Database server has the following types of values for initialization parameters:

- Boolean
- String
- Integer
- Parameter File
- Reserved
- Big Integer

Derived Parameter Values

Some initialization parameters are derived, meaning that their values are calculated from the values of other parameters. Normally, you should not alter values for derived parameters. But if you do, the value that you specify overrides the calculated value.

For example, the default value of the SESSIONS parameter is derived from the value of the PROCESSES parameter. If the value of PROCESSES changes, the default value of SESSIONS changes as well, unless you override it with a specified value.

Operating System–Dependent Parameter Values

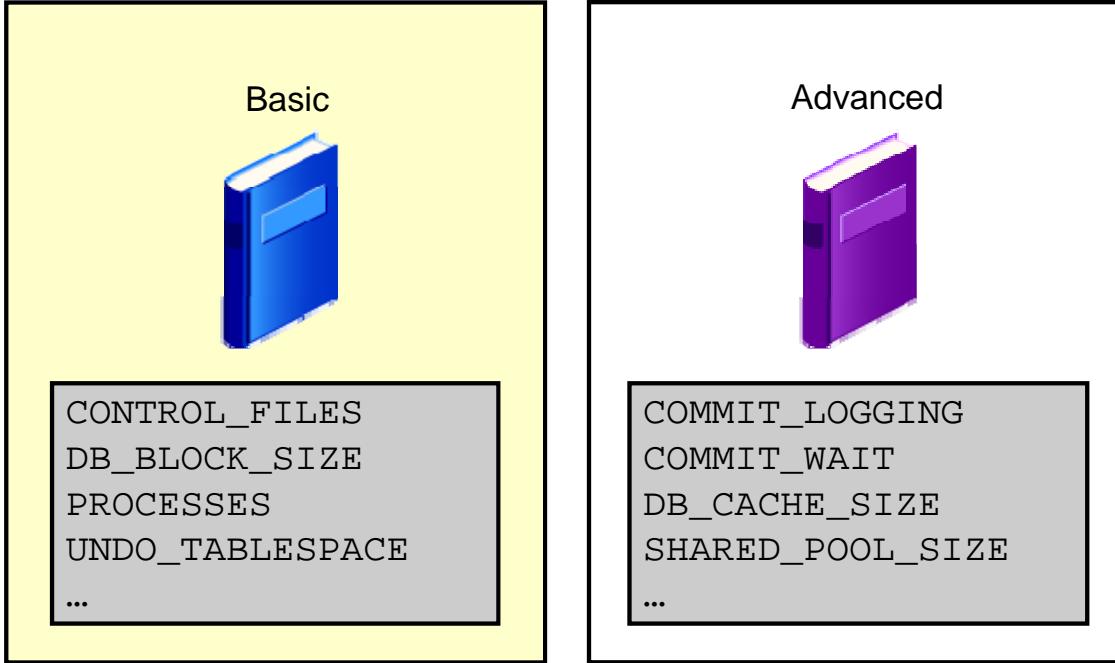
The valid values or value ranges of some initialization parameters depend on the host operating system. For example, the DB_FILE_MULTIBLOCK_READ_COUNT parameter specifies the maximum number of blocks that are read in one I/O operation during a sequential scan; this parameter is platform dependent. The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

Setting Parameter Values

Initialization parameters offer the most potential for improving system performance. Some parameters set capacity limits but do not affect performance. For example, when the value of OPEN_CURSORS is 10, a user process attempting to open its eleventh cursor receives an error. Other parameters affect performance but do not impose absolute limits. For example, reducing the value of OPEN_CURSORS does not prevent work even though it may slow performance.

Increasing the values of parameters may improve your system's performance, but increasing most parameters also increases the System Global Area (SGA) size. A larger SGA can improve database performance up to a point. An SGA that is too large can degrade performance if it is swapped in and out of memory. Operating system parameters that control virtual memory working areas should be set with the SGA size in mind. The operating system configuration can also limit the maximum size of the SGA.

Types of Initialization Parameters



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Initialization parameters are of two types: basic and advanced.

In the majority of cases, it is necessary to set and tune only the 30 or so basic parameters to get reasonable performance from the database. In rare situations, modification of the advanced parameters may be needed to achieve optimal performance. There are more than 300 advanced parameters.

A basic parameter is defined as one that you are likely to set to keep your database running with good performance. All other parameters are considered to be advanced.

Examples of basic parameters:

- Determining the global database name: DB_NAME and DB_DOMAIN
- Specifying a fast recovery area and size: DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE
- Specifying the total size of all SGA components: SGA_TARGET
- Specifying the method of undo space management tablespace: UNDO_TABLESPACE
- COMPATIBLE initialization parameter and irreversible compatibility

Note: Some of the initialization parameters are listed on the following pages. For a complete list, see the *Oracle Database Reference*.

Initialization Parameters: Examples

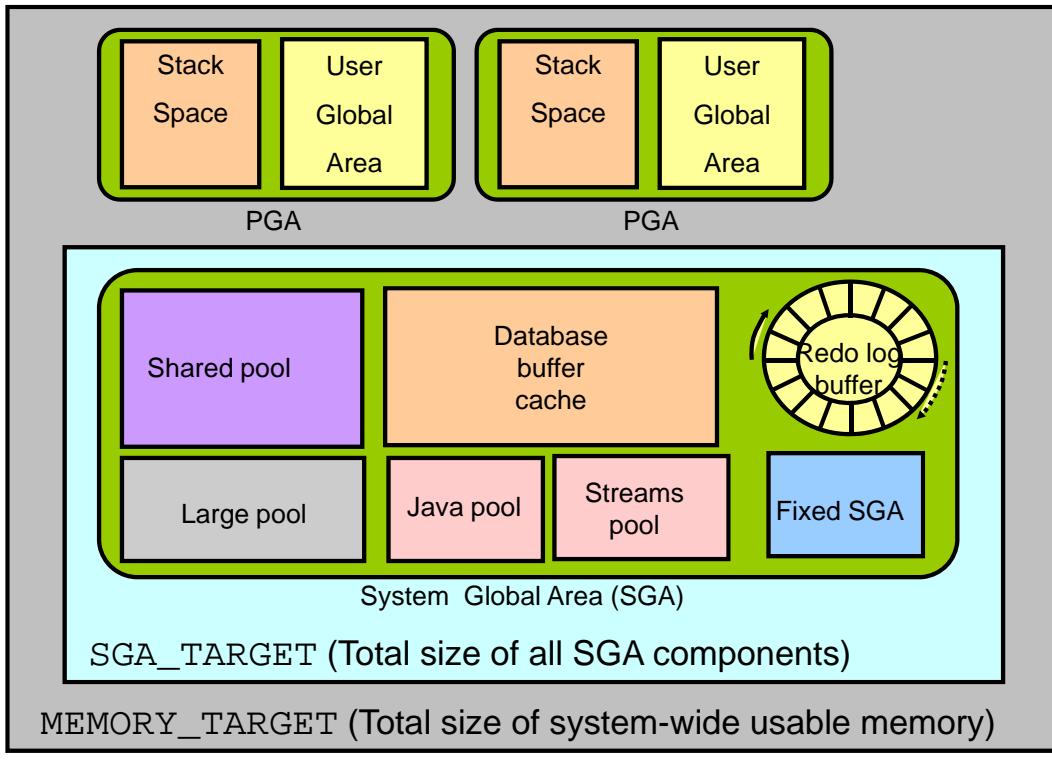
Parameter	Specifies
CONTROL_FILES	One or more control file names
DB_FILES	Maximum number of database files
PROCESSES	Maximum number of OS user processes that can simultaneously connect
DB_BLOCK_SIZE	Standard database block size used by all tablespaces
DB_CACHE_SIZE	Size of the standard block buffer cache



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **CONTROL_FILES parameter:** Specifies one or more control file names. Oracle strongly recommends that you multiplex and mirror control files. Range of values: from one to eight file names (with path names). Default value: OS dependent.
- **DB_FILES parameter:** Specifies the maximum number of database files that can be opened for this database. Range of values: OS dependent. Default value: 200.
- **PROCESSES parameter:** Specifies the maximum number of OS user processes that can simultaneously connect to an Oracle server. This value should allow for all background processes and user processes. Range of values: from 6 to an OS-dependent value. Default value: Dynamic and dependent on the number of CPUs.
- **DB_BLOCK_SIZE parameter:** Specifies the size (in bytes) of an Oracle database block. This value is set at database creation and cannot be subsequently changed. This specifies the standard block size for the database. All tablespaces will use this size by default. Range of values: 2048 to 32768 (OS-dependent). Default value: 8192.
- **DB_CACHE_SIZE parameter:** Specifies the size of the default buffer pool. Range of values: At least 4 MB times the number of CPUs (smaller values are automatically rounded up to this value). Default value: 0 if SGA_TARGET is set, otherwise the larger of 48 MB or (4 MB * CPU_COUNT).

Initialization Parameters: Examples



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

SGA_TARGET specifies the total size of all SGA components. If **SGA_TARGET** is specified, the following memory pools are automatically sized:

- Buffer cache (`DB_CACHE_SIZE`)
- Shared pool (`SHARED_POOL_SIZE`)
- Large pool (`LARGE_POOL_SIZE`)
- Java pool (`JAVA_POOL_SIZE`)
- Streams pool (`STREAMS_POOL_SIZE`)

If these automatically tuned memory pools are set to nonzero values, the values are used as minimum levels by Automatic Shared Memory Management (ASMM). You set minimum values if an application component needs a minimum amount of memory to function properly.

The following pools are manually sized components and are not affected by ASMM:

- Log buffer
- Other buffer caches (such as `KEEP` and `RECYCLE`) and other block sizes
- Fixed SGA and other internal allocations

The memory allocated to these pools is deducted from the total available memory for **SGA_TARGET** when ASMM is enabled.

Note: The MMON process computes the values of the automatically tuned memory pools to support ASMM.

MEMORY_TARGET specifies the Oracle systemwide usable memory. The database tunes memory to the MEMORY_TARGET value, reducing or enlarging the SGA and PGA as needed.

In a text-based initialization parameter file, if you omit MEMORY_MAX_TARGET and include a value for MEMORY_TARGET, the database automatically sets MEMORY_MAX_TARGET to the value of MEMORY_TARGET. If you omit the line for MEMORY_TARGET and include a value for MEMORY_MAX_TARGET, the MEMORY_TARGET parameter defaults to zero. After startup, you can then dynamically change MEMORY_TARGET to a nonzero value if it does not exceed the value of MEMORY_MAX_TARGET. The MEMORY_TARGET parameter is modifiable with the ALTER SYSTEM command. Values range from 152 MB to MEMORY_MAX_TARGET.

Initialization Parameters: Examples

Parameter	Specifies
PGA_AGGREGATE_TARGET	Amount of PGA memory available to all server processes
SHARED_POOL_SIZE	Size of shared pool (in bytes)
UNDO_MANAGEMENT	Undo space management mode to be used



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **PGA_AGGREGATE_TARGET parameter:** Specifies the amount of Program Global Area (PGA) memory available to all server processes attached to the instance. This memory does not reside in the System Global Area (SGA). The database uses this parameter as a target amount of PGA memory to use. When setting this parameter, subtract the SGA from the total memory on the system that is available to the Oracle instance. The minimum value is 10 MB and the maximum value is (4096 GB – 1). The default is 10 MB or 20% of the size of the SGA, whichever is greater.
- **SHARED_POOL_SIZE parameter:** Specifies the size of the shared pool in bytes. The shared pool contains objects such as shared cursors, stored procedures, control structures, and parallel execution message buffers. Range of values: OS-dependent. Default value: 0 if SGA_TARGET is set, otherwise 128 MB if 64-bit; 48 MB if 32-bit.
- **UNDO_MANAGEMENT parameter:** Specifies the undo space management mode that the system should use. When set to AUTO, the instance is started in automatic undo management mode. Otherwise, it is started in rollback undo mode. In rollback undo mode, undo space is allocated as rollback segments. In automatic undo mode, undo space is allocated as undo tablespaces. Range of values: AUTO or MANUAL. If the UNDO_MANAGEMENT parameter is omitted when the instance is started, the default value AUTO is used.

Using SQL*Plus to View Parameters

```
SQL> SELECT name, value FROM v$parameter;
NAME          VALUE
-----
lock_name_space
processes      300
sessions       472
timed_statistics  TRUE
timed_os_statistics 0
...
SQL> SHOW PARAMETER SHARED_POOL_SIZE
NAME          TYPE          VALUE
-----
shared_pool_size  big integer 0
SQL> show parameter para
NAME          TYPE          VALUE
-----
cell_offload_parameters  string
fast_start_parallel_rollback  string      LOW
parallel_adaptive_multi_user  boolean     TRUE
parallel_automatic_tuning    boolean     FALSE
...
...
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The slide shows examples of using SQL*Plus to view parameters. You can query the V\$PARAMETER view to find the values of the various parameters. V\$PARAMETER displays the current parameter values in the current session. You can also use the SHOW PARAMETER command with any string to view parameters that contain that string.

The query in the following example is requesting the name and values of the parameters. Use a WHERE clause to specify specific parameter names:

```
SQL> SELECT name, value FROM v$parameter
      2 WHERE name LIKE '%pool%';
NAME          VALUE
-----
shared_pool_size 0
large_pool_size 0
java_pool_size 0
streams_pool_size 0
shared_pool_reserved_size 15728640
...
9 rows selected.
```

Description of the view:

```
SQL> desc V$parameter
```

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

The second example shows the use of the SQL*Plus SHOW PARAMETER command to view parameter settings. You can also use this command to find all parameters that contain a text string. For example, you can find all parameter names that include the db string by using the following command:

```
SQL> show parameter db
```

NAME	TYPE	VALUE
..		
db_8k_cache_size	big integer	0
db_big_table_cache_percent_target	string	0
db_block_buffers	integer	0
db_block_checking	string	FALSE
db_block_checksum	string	TYPICAL
..		

Other Views Containing Information About Parameters

- V\$SPPARAMETER: Displays information about the contents of the server parameter file. If a server parameter file was not used to start the instance, each row of the view will contain FALSE in the ISSPECIFIED column.
- V\$PARAMETER2: Displays information about the initialization parameters that are currently in effect for the session, with each parameter value appearing as a row in the view. A new session inherits parameter values from the instance-wide values displayed in the V\$SYSTEM_PARAMETER2 view.
- V\$SYSTEM_PARAMETER: Displays information about the initialization parameters that are currently in effect for the instance

Changing Initialization Parameter Values

- Static parameters:
 - Can be changed only in the parameter file
 - Require restarting the instance before taking effect
- Dynamic parameters:
 - Can be changed while database is online
 - Can be altered at:
 - Session level
 - System level
 - Are valid for duration of session or based on SCOPE setting
 - Are changed by using the ALTER SESSION and ALTER SYSTEM commands



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are two types of initialization parameters.

Static parameters: Affect the instance or entire database and can be modified only by changing the contents of the text initialization parameter file or the server parameter file. Static parameters require the database to be shut down and restarted to take effect. They cannot be changed for the current instance.

Dynamic parameters: Can be changed while the database is online. There are two types:

- *Session-level parameters* affect only a user session. Examples include national language support (NLS) parameters that can be used to specify national language settings for sorts, date parameters, and so on. You can use these in a given session; they expire when the session ends.
- *System-level parameters* affect the entire database and all sessions. Examples include modifying the SGA_TARGET value and setting archive log destinations. These parameters stay in effect based on the SCOPE specification. To make them permanent, you have to add these parameter settings to the server parameter file by specifying the SCOPE=both option or by manually editing the text initialization parameter file.

Dynamic parameters can be changed by using the ALTER SESSION and ALTER SYSTEM commands.

Use the `SET` clause of the `ALTER SYSTEM` statement to set or change initialization parameter values. The optional `SCOPE` clause specifies the scope of a change as follows:

- **SCOPE=SPFILE:** The change is applied in the server parameter file only. No change is made to the current instance. For both dynamic and static parameters, the change is effective at the next startup and is persistent. This is the only `SCOPE` specification allowed for static parameters.
- **SCOPE=MEMORY:** The change is applied in memory only. The change is made to the current instance and is effective immediately. For dynamic parameters, the effect is immediate but not persistent because the server parameter file is not updated. For static parameters, this specification is not allowed.
- **SCOPE=BOTH:** The change is applied in both the server parameter file and memory. The change is made to the current instance and is effective immediately. For dynamic parameters, the effect is persistent because the server parameter file is updated. For static parameters, this specification is not allowed.

You must not specify `SCOPE=SPFILE` or `SCOPE=BOTH` if the instance did not start up with a server parameter file. The default is `SCOPE=BOTH` if a server parameter file was used to start up the instance, and the default is `MEMORY` if a text initialization parameter file was used to start up the instance.

For some dynamic parameters, you can also specify the `DEFERRED` keyword. When it is specified, the change is effective only for future sessions. This is valid for only the following parameters:

- `AUDIT_FILE_DEST`
- `BACKUP_TAPE_IO_SLAVES`
- `OBJECT_CACHE_MAX_SIZE_PERCENT`
- `OBJECT_CACHE_OPTIMAL_SIZE`
- `OLAP_PAGE_POOL_SIZE`
- `RECYCLEBIN`
- `SORT_AREA_RETAINED_SIZE`
- `SORT_AREA_SIZE`

When you specify `SCOPE` as `SPFILE` or as `BOTH`, an optional `COMMENT` clause lets you associate a text string with the parameter update. The comment is written to the server parameter file.

Changing Parameter Values: Examples

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

Session altered.

```
SQL> SELECT SYSDATE FROM dual;
```

SYSDATE

oct 17 2012

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

System altered.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The first statement in the slide is an example of changing a session-level parameter. The user is setting the session date format to be `mon dd yyyy`. As a result, any queries on the date will display dates in that format. Session-level parameters can also be set in applications by using PL/SQL.

The second statement changes the maximum number of failed login attempts before the connection is dropped. It includes a comment and explicitly states that the change is to be made only in the server parameter file. After the specified number of failure attempts, the connection is automatically dropped by the server process. This is not a dynamic parameter and the Oracle database instance will need to be restarted before the change can take effect.

Quiz

The majority of database parameters are dynamic and can be changed without having to shut down the database instance.

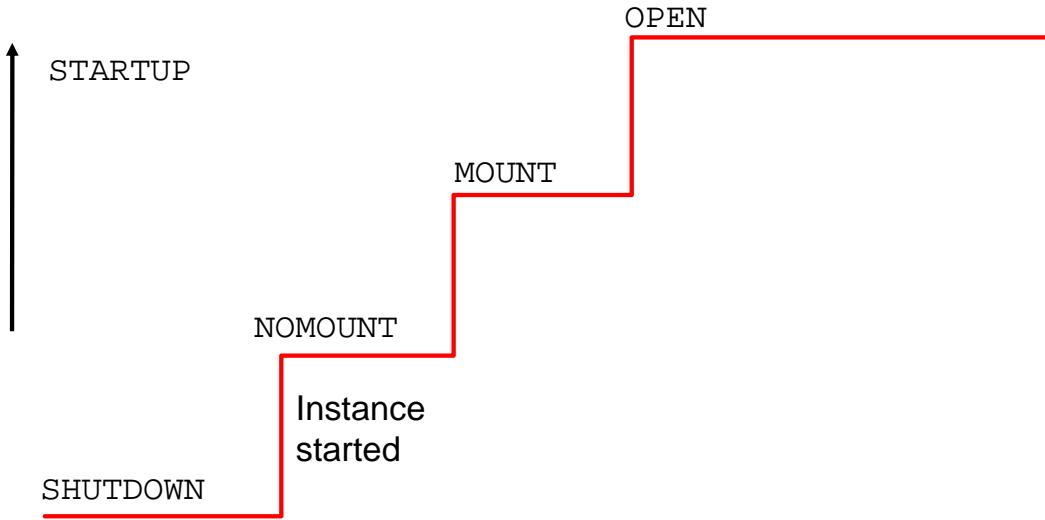
- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Starting Up an Oracle Database Instance: NOMOUNT



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The database instance and database go through stages as the database is made available for access by users. The database instance is started, the database is mounted, and then the database is opened.

An instance is typically started only in NOMOUNT mode during database creation, during re-creation of control files, or in certain backup and recovery scenarios.

When an instance is started, the following takes place:

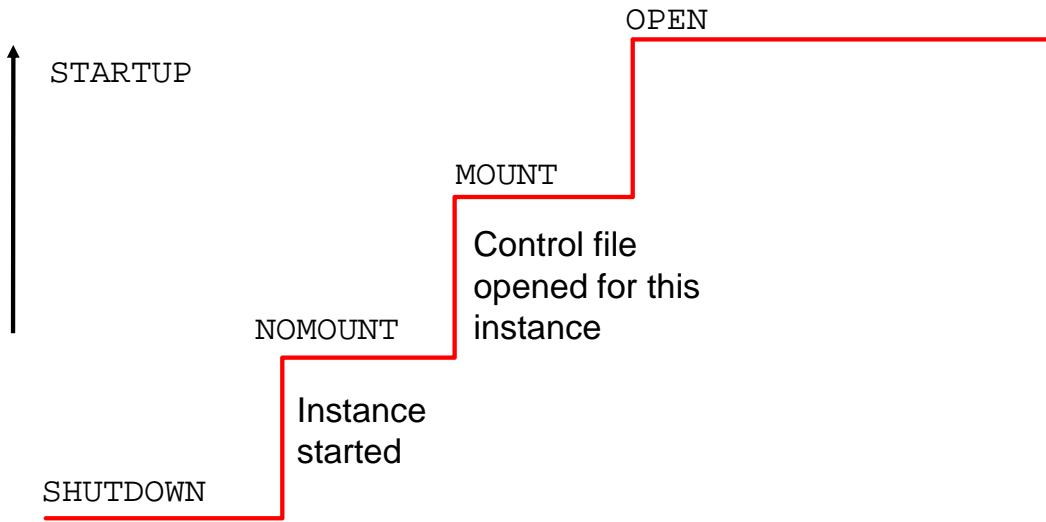
- Searching \$ORACLE_HOME/dbs for a file of a particular name in this sequence:
 1. Search for spfile<SID>.ora.
 2. If spfile<SID>.ora is not found, search for spfile.ora.
 3. If spfile.ora is not found, search for init<SID>.ora.

This is the file that contains initialization parameters for the instance. Specifying the PFILE parameter with STARTUP overrides the default behavior.

- Allocating the SGA
- Starting the background processes
- Opening the alert_<SID>.log file and the trace files

Note: SID is the system ID, which identifies the instance name (for example, ORCL).

Starting Up an Oracle Database Instance: MOUNT



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Mounting a database includes the following:

- Associating a database with a previously started instance
- Locating and opening all the control files specified in the parameter file
- Reading the control files to obtain the names and statuses of the data files and online redo log files. (However, no checks are performed to verify the existence of the data files and online redo log files at this time.)

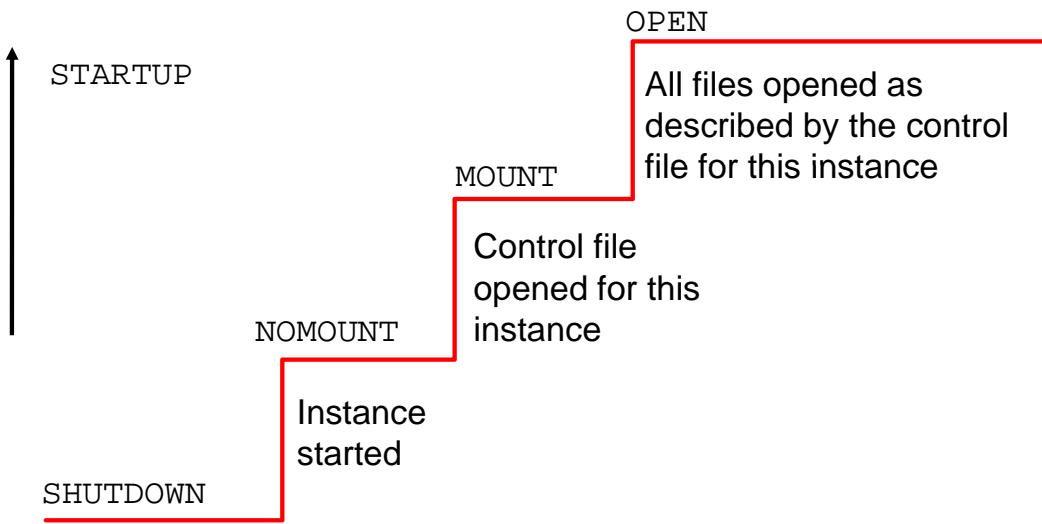
To perform specific maintenance operations, start an instance and mount a database, but do not open the database.

For example, the database must be mounted but must not be opened during the following tasks:

- Renaming data files. (Data files for an offline tablespace can be renamed when the database is open.)
- Enabling and disabling online redo log file archiving options
- Performing full database recovery

Note: A database may be left in MOUNT mode even though an OPEN request has been made. This may be because the database needs to be recovered in some way. If recovery is performed while in the MOUNT state, the redo logs are open for reads and the data files are open as well to read the blocks needing recovery and to write blocks if required during recovery.

Starting Up an Oracle Database Instance: OPEN



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A normal database operation means that an instance is started and the database is mounted and opened. With a normal database operation, any valid user can connect to the database and perform typical data access operations.

Opening the database includes the following:

- Opening the data files
- Opening the online redo log files

If any of the data files or online redo log files are not present when you attempt to open the database, the Oracle server returns an error.

During this final stage, the Oracle server verifies that all data files and online redo log files can be opened, and checks the consistency of the database. If necessary, the System Monitor (SMON) background process initiates instance recovery.

You can start up a database instance in restricted mode so that only Oracle Database users with the RESTRICTED SESSION system privilege can connect to the database.

Startup Options: Examples

- Using the SQL*Plus utility:

```
SQL> startup
```

1

```
SQL> startup nomount
```

2

```
SQL> alter database mount;
```

3

```
SQL> alter database open;
```

4

- Using the Server Control utility with Oracle Restart

```
$ srvctl start database -d orcl -o mount
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The slide shows the SQL*Plus syntax to start up the database.

- This command starts the instance, associates the database files to it, and mounts and opens the database.
- This command starts the instance and the database is not mounted.
- This command mounts a database from the NOMOUNT state.
- This command opens the database from the MOUNT state.

When the database is enabled with Oracle Restart, the Server Control (SRVCTL) utility can be used to start the database instance. The SRVCTL utility has the advantage that it can also start all required dependent resources such as the ASM instance, ASM disk groups, and listener.

Shutdown Modes

Shutdown Modes	A	I	T	N
Allows new connections	No	No	No	No
Waits until current sessions end	No	No	No	Yes
Waits until current transactions end	No	No	Yes	Yes
Forces a checkpoint and closes files	No	Yes	Yes	Yes

Shutdown modes:

- A = ABORT
- I = IMMEDIATE
- T = TRANSACTIONAL
- N = NORMAL

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Shutdown modes are progressively more accommodating of current activity in this order:

- **ABORT:** Performs the least amount of work before shutting down. Because this mode requires recovery before startup, use it only when necessary. It is typically used when no other form of shutdown works, when there are problems with starting the instance, or when you need to shut down immediately because of an impending situation (such as notice of a power outage within seconds).
- **IMMEDIATE:** Is the most typically used option. Uncommitted transactions are rolled back.
- **TRANSACTIONAL:** Allows existing transactions to finish, but does not allow new transactions to start
- **NORMAL:** Waits for sessions to disconnect

If you consider the amount of time that it takes to perform the shutdown, you find that **ABORT** is the fastest and **NORMAL** is the slowest. **NORMAL** and **TRANSACTIONAL** can take a long time depending on the number of sessions and transactions.

Shutdown Options

On the way down:

- Uncommitted changes rolled back, for IMMEDIATE
- Database buffer cache written to data files
- Resources released

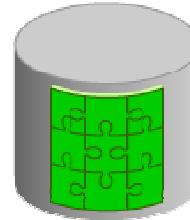
During:

SHUTDOWN
NORMAL
or
SHUTDOWN
TRANSACTIONAL
or
SHUTDOWN
IMMEDIATE

On the way up:

- No instance recovery

Consistent database



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

SHUTDOWN NORMAL

NORMAL is the default shutdown mode if no mode is specified. A normal database shutdown proceeds with the following conditions:

- No new connections can be made.
- The Oracle server waits for all users to disconnect before completing the shutdown.
- Database and redo buffers are written to disk.
- Background processes are terminated and the SGA is removed from memory.
- The Oracle server closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.

SHUTDOWN TRANSACTIONAL

A shutdown in TRANSACTIONAL mode prevents clients from losing data, including results from their current activity. A transactional database shutdown proceeds with the following conditions:

- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have been completed, a shutdown occurs immediately.
- The next startup does not require an instance recovery.

SHUTDOWN IMMEDIATE

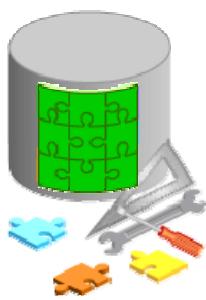
A shutdown in `IMMEDIATE` mode proceeds with the following conditions:

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

Shutdown Options

On the way down:

- Modified buffers not written to data files
- Uncommitted changes not rolled back



During:

SHUTDOWN ABORT
or
Instance failure
or
STARTUP FORCE

Inconsistent database

On the way up:

- Online redo log files used to reapply changes
- Undo segments used to roll back uncommitted changes
- Resources released

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

SHUTDOWN ABORT

If shutdown in NORMAL, TRANSACTIONAL, and IMMEDIATE modes does not work, you can abort the current database instance. Aborting an instance proceeds with the following conditions:

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

Note: It is not advisable to back up a database that is in an inconsistent state.

Shutdown Options: Examples

- Using SQL*Plus:

```
SQL> shutdown
```

1

```
SQL> shutdown transactional
```

2

```
SQL> shutdown immediate
```

3

```
SQL> shutdown abort
```

4

- Using the SRVCTL utility with Oracle Restart

```
$ srvctl stop database -d orcl -o abort
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The slide shows examples using both SQL*Plus and the SRVCTL utility to shut down the database.

1. This command initiates a normal shutdown. The database will not shut down until all users have logged out.
2. This command initiates a transactional shutdown. The database will not shut down until all existing transactions are completed.
3. This command initiates an immediate shutdown. Uncommitted transactions will be rolled back.
4. This command initiates a shutdown abort.

When the database is enabled with Oracle Restart, the SRVCTL utility can be used to shut down the database instance.

Viewing the Alert Log

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. On the left, the navigation menu is open, with 'Logs' selected. A sub-menu is displayed under 'Logs', showing 'Alert Log Contents' highlighted. To the right, there is a chart titled 'Performance Activity Class' showing 'Active Sessions' over time. Below the chart, a red box highlights a message from the alert log:

```

Current log# 3 seq# 27 mem# 0: /u01/app/oracle/oradata/orcl/redo03.log
Mon Oct 06 21:07:54 2014
Resize operation completed for file# 3, old size 849920K, new size 860160K
Mon Oct 06 22:00:01 2014
Setting Resource Manager plan SCHEDULER[0x4443]:DEFAULT_MAINTENANCE_PLAN via scheduler window
Setting Resource Manager plan DEFAULT_MAINTENANCE_PLAN via parameter
Mon Oct 06 22:00:03 2014
Begin automatic SQL Tuning Advisor run for special tuning task "SYS_AUTO_SQL_TUNING_TASK"
End automatic SQL Tuning Advisor run for special tuning task "SYS_AUTO_SQL_TUNING_TASK"

```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

The alert file of a database is a chronological log of messages such as the following:

- Any nondefault initialization parameters used at startup
- All internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60) that occurred
- Administrative operations, such as the SQL statements CREATE, ALTER, DROP DATABASE, and TABLESPACE; and the Enterprise Manager or SQL*Plus statements STARTUP, SHUTDOWN, ARCHIVE LOG, and RECOVER
- Several messages and errors relating to the functions of shared server and dispatcher processes
- Errors during the automatic refresh of a materialized view

Oracle Database uses the alert log to keep a record of these events as an alternative to displaying the information on an operator's console. (Many systems also display this information on the console.) If an administrative operation is successful, a message is written in the alert log as "completed" along with a time stamp.

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

Note: There is an XML version of the alert log in the \$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/alert directory.

To determine the location of the alert log with SQL*Plus:

- Connect to the database with SQL*Plus (or another query tool such as SQL Developer).
- Query the V\$DIAG_INFO view.

To view the text-only alert log without the XML tags:

- In the V\$DIAG_INFO query results, note the path that corresponds to the Diag Trace entry. Change to the location specified.
- Open the alert_SID.log file with a text editor.

To view the XML-formatted alert log:

- In the V\$DIAG_INFO query results, note the path that corresponds to the Diag Alert entry. Change directory to that path.
- Open the log.xml file with a text editor.

Using Trace Files

- Each server and background process can write to an associated trace file.
- Error information is written to the corresponding trace file.
- Automatic diagnostic repository (ADR)
 - Is a systemwide central tracing and logging repository
 - Stores database diagnostic data such as:
 - Traces
 - Alert log
 - Health monitor reports



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each server and background process can write to an associated trace file. When a process detects an internal error, it dumps information about the error to its trace file. If an internal error occurs and information is written to a trace file, the administrator should contact Oracle Support Services.

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

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

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

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

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

The ADR root directory is known as *ADR base*. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter.

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

```
./diag/product_type/db_id/instance_id
```

Administering the DDL Log File

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

```
$ more ddl_orcl.log
Thu Nov 15 08:35:47 2012
diag_adl:drop user app_user
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The DDL log is created only if the `ENABLE_DDL_LOGGING` initialization parameter is set to `TRUE`. When this parameter is set to `FALSE`, DDL statements are not included in any log. A subset of executed DDL statements is written to the DDL log. Refer to the *Oracle Database Reference* for a complete list of DDL commands that are captured in the DDL log.

Locate the log files as follows:

```
$ pwd
/u01/app/oracle/diag/rdbms/orcl/orcl/log
$ ls
ddl  ddl_orcl.log  debug  test
$ cd ddl
$ ls
log.xml
```

Understanding the Debug Log File

- Debug log contains warnings about conditions, states, or events that do not inhibit correct operation of an Oracle Database component.
- The log is intended for use by Oracle Support when diagnosing a problem.
- It is included in incident packaging service (IPS) incident packages.
- It is written to
\$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/debug.

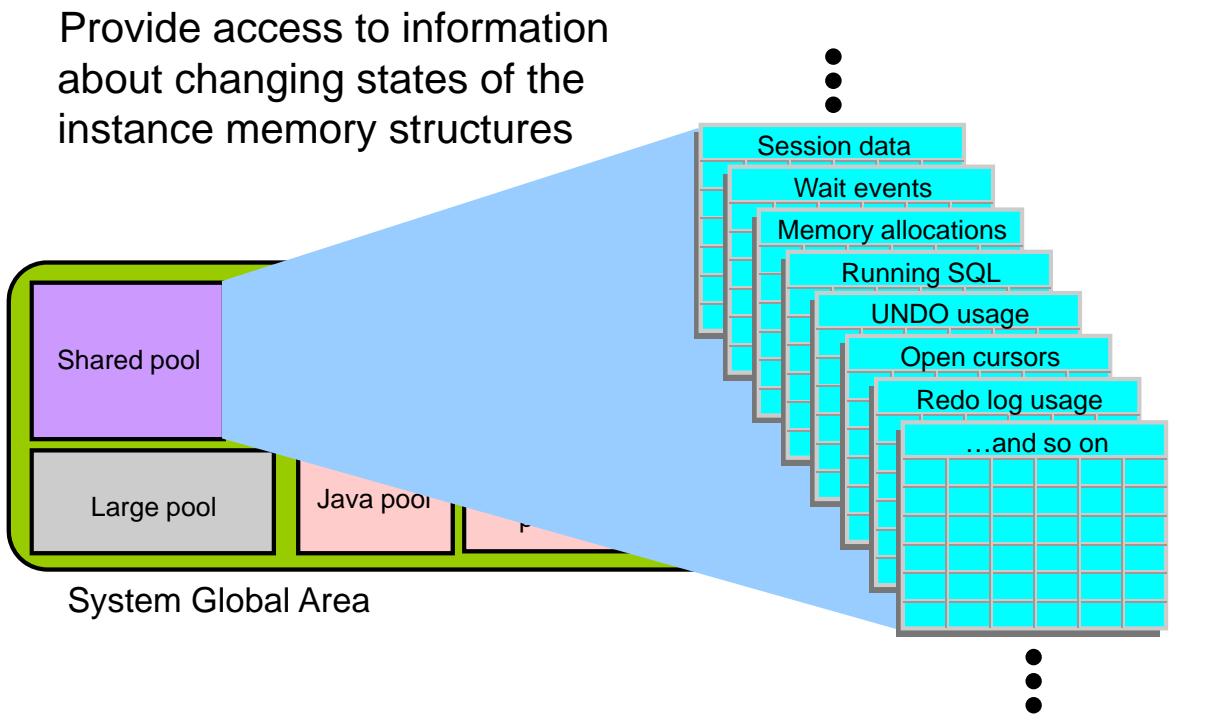


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The debug log contains warnings generated by Oracle Database components. The warnings are about conditions, states, or events that may be out of the ordinary but do prevent the component from operating correctly. For this reason, these warnings are not written to the alert log. If they are needed in the future to diagnose a problem, they are recorded in the debug log.

Typically, the debug log would be needed only by Oracle Support to diagnose a problem, so you do not need to view the contents of the debug log on a regular basis. The debug log is also included in incident packaging service (IPS) incident packages. IPS is discussed in detail in a later lesson.

Using Dynamic Performance Views



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The Oracle Database server also maintains a more dynamic set of data about the operation and performance of the database instance. These dynamic performance views are based on virtual tables that are built from memory structures inside the database server. That is, they are not conventional tables that reside in a database. This is the reason why some of them are available before a database is mounted or open.

Dynamic performance views include information about:

- Sessions
- File states
- Progress of jobs and tasks
- Locks
- Backup status
- Memory usage and allocation
- System and session parameters
- SQL execution
- Statistics and metrics

Note: The `DICT` and `DICT_COLUMNS` views also contain the names of these dynamic performance views. Dynamic performance views start with the prefix `v$`.

Dynamic Performance Views: Usage Examples

1

```
SELECT sql_text, executions FROM v$sql  
WHERE cpu_time > 200000;
```

2

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

3

```
SELECT sid, ctime FROM v$lock  
WHERE block > 0;
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The three examples shown in the slide answer the following questions:

1. For which SQL statements (and their associated numbers of executions) is the CPU time consumed greater than 200,000 microseconds?
2. Which current sessions have logged in from the EDXX9P1 computer in the last day?
3. What are the session IDs of those sessions that are currently holding a lock that is blocking another user, and how long have those locks been held?

Dynamic Performance Views: Considerations

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

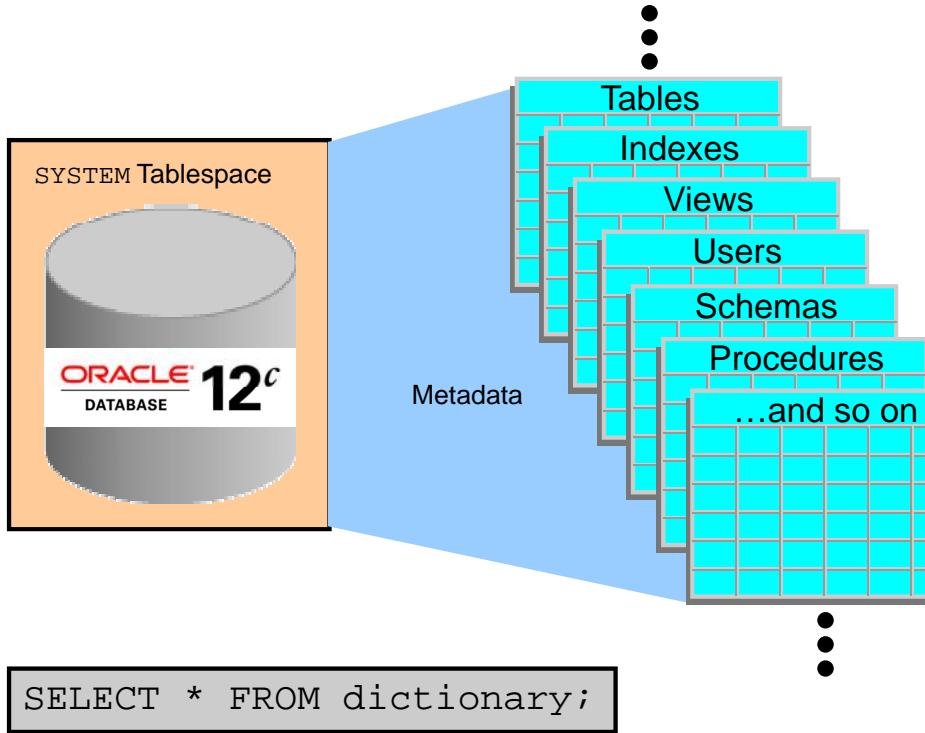


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Some dynamic views contain data that is not applicable to all states of an instance or database. For example, if an instance has just been started but no database is mounted, you can query `V$BGPROCESS` to see the list of background processes that are running. But you cannot query `V$DATAFILE` to see the status of database data files because it is the mounting of a database that reads the control file to find out about the data files associated with a database.

Some `V$` views contain information that is similar to information in the corresponding `DBA_` views. For example, `V$DATAFILE` is similar to `DBA_DATA_FILES`. Note also that `V$` view names are generally singular and `DBA_` view names are plural.

Data Dictionary: Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

The data dictionary:

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

Note: The `DICTIONARY` data dictionary view (or the `DICT` synonym for this) contains the names and descriptions of data dictionary tables and views. Use the `DICT_COLUMNS` view to see the view columns and their definitions. For complete definitions of each view, see the *Oracle Database Reference*. There are over 1000 views that reference hundreds of base tables.

Data Dictionary Views

	Who Can Query	Contents	Subset of	Notes
DBA_	DBA	Everything	N/A	May have additional columns meant for DBA use only
ALL_	Everyone	Everything that the user has privileges to see	DBA_ views	Includes user's own objects and other objects that the user has been granted privileges to see
USER_	Everyone	Everything that the user owns	ALL_ views	Is usually the same as ALL_ except for the missing OWNER column. (Some views have abbreviated names as PUBLIC synonyms.)



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The view prefixes indicate the data (and how much of that data) a given user can see.

A global view of everything is accessed only by users with DBA privileges, using the DBA_ prefix.

The next level of privilege is at the ALL_ prefix level, which represents all objects that the querying user is privileged to see, whether the user owns them or not. For example, if USER_A has been granted access to a table owned by USER_B, then USER_A sees that table listed in any ALL_ view dealing with table names.

The USER_ prefix represents the smallest scope of visibility. This type of view shows only those objects that the querying user owns (that is, those that are present in the user's own schema).

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

The DBA_ views can be queried only by users with the SYSDBA, SELECT ANY DICTIONARY, or SELECT_CATALOG_ROLE privilege.

Not all dictionary views start with the prefix DBA_, ALL_, and USER_. The following views or synonyms to views are exceptions to this:

- AUDIT_ACTIONS
- CAT
- CHANGE_PROPAGATIONS
- CHANGE_PROPAGATION_SETS
- CHANGE_SETS
- CHANGE_SOURCES
- CHANGE_TABLES
- CLIENT_RESULT_CACHE_STATS\$
- CLU
- COLS
- COLUMN_PRIVILEGES
- DATABASE_COMPATIBLE_LEVEL
- DBMS_ALERT_INFO
- DBMS_LOCK_ALLOCATED
- DICT
- DICTIONARY
- DICT_COLUMNS
- DUAL
- GLOBAL_NAME
- IND
- INDEX_HISTOGRAM
- INDEX_STATS
- LOGSTDBY_UNSUPPORTED_TABLES
- NLS_DATABASE_PARAMETERS
- NLS_INSTANCE_PARAMETERS
- NLS_SESSION_PARAMETERS
- OBJ
- RECYCLEBIN
- RESOURCE_COST
- ROLE_ROLE_PRIVS
- ROLE_SYS_PRIVS
- ROLE_TAB_PRIVS
- SEQ
- SESSION_PRIVS
- SESSION_ROLES

- SM\$VERSION
- SYN
- TABLE_PRIVILEGES
- TABS

Data Dictionary: Usage Examples

1

```
SELECT table_name, tablespace_name  
FROM user_tables;
```

2

```
SELECT sequence_name, min_value, max_value,  
increment_by  
FROM all_sequences  
WHERE sequence_owner IN ('MDSYS', 'XDB');
```

3

```
SELECT USERNAME, ACCOUNT_STATUS  
FROM dba_users  
WHERE ACCOUNT_STATUS = 'OPEN';
```

4

```
DESCRIBE dba_indexes
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The example queries in the slide answer the following questions:

1. What are the names of the tables (along with the name of the tablespace where they reside) that have been created in your schema?
2. What is the significant information about the sequences in the database that you have access to?
3. What users in this database are currently able to log in?
4. What are the columns of the DBA_INDEXES view? This shows you what information you can view about all the indexes in the database. The following is a partial output of this command:

```
SQL> DESCRIBE dba_indexes  
Name          Null?    Type  
-----  
OWNER          NOT NULL VARCHAR2( 30 )  
INDEX_NAME     NOT NULL VARCHAR2( 30 )  
INDEX_TYPE      VARCHAR2( 27 )  
TABLE_OWNER     NOT NULL VARCHAR2( 30 )  
TABLE_NAME      NOT NULL VARCHAR2( 30 )
```

Quiz

Which data dictionary view can be used to find the names of all tables in the database?

- a. USER_TABLES
- b. ALL_TABLES
- c. DBA_TABLES
- d. ANY_TABLES



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Start and stop the Oracle database instance and components
- Modify database initialization parameters
- Describe the stages of database startup
- Describe database shutdown options
- View the alert log
- Access dynamic performance views



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Viewing and modifying initialization parameters
- Stopping and starting the database instance
- Viewing log files



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Configuring the Oracle Network Environment



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

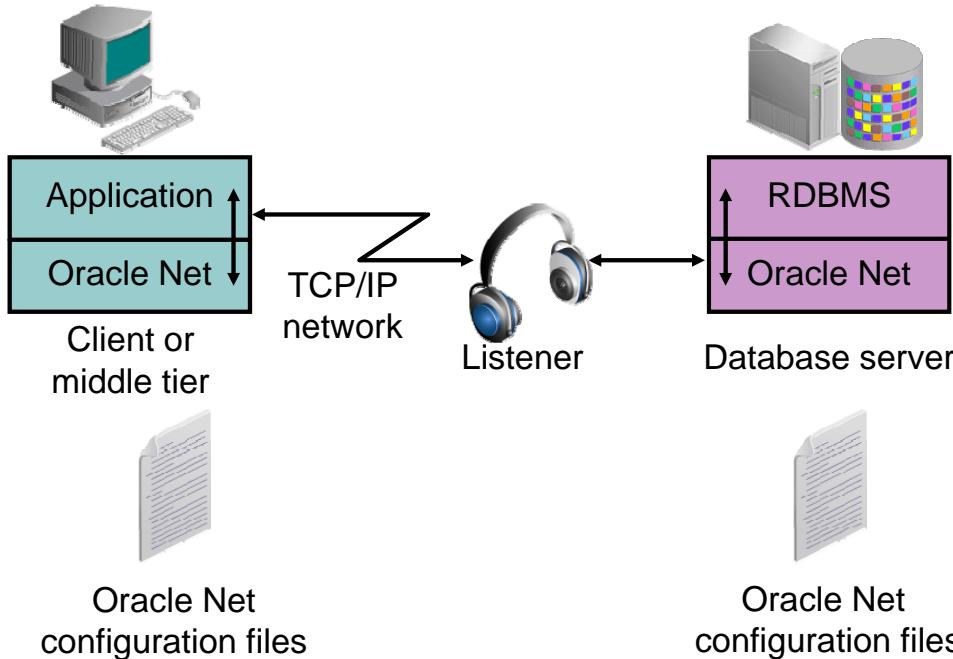
After completing this lesson, you should be able to:

- Use Enterprise Manager Cloud Control and Oracle Net Manager to:
 - Create additional listeners
 - Create Oracle Net Service aliases
 - Control Oracle Net Listener
- Use the Listener Control Utility to manage Oracle Net Listener
- Use `tnsping` to test Oracle Net connectivity
- Identify when to use shared servers and when to use dedicated servers



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Net Services: Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

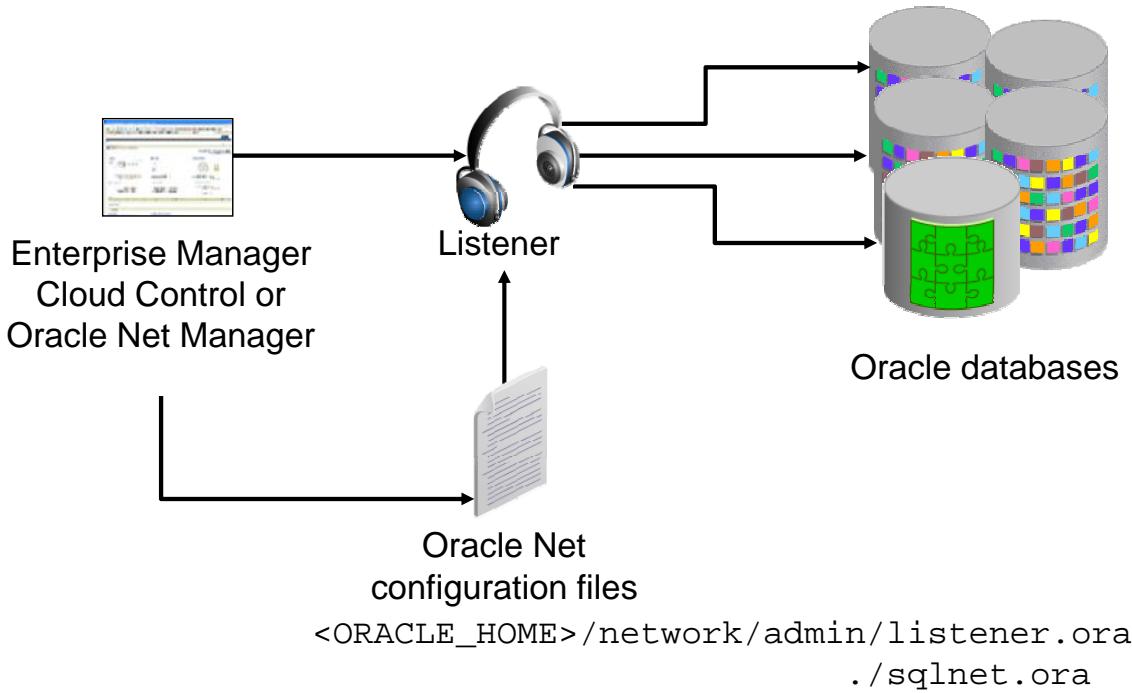
Oracle Net Services enables network connections from a client or middle-tier application to the Oracle server. After a network session is established, Oracle Net acts as the data courier for both the client application and the database server. It is responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. Oracle Net (or something that simulates Oracle Net, such as Java Database Connectivity) is located on each computer that needs to talk to the database server.

On the client computer, Oracle Net is a background component for application connections to the database.

On the database server, Oracle Net includes an active process called *Oracle Net Listener*, which is responsible for coordinating connections between the database and external applications.

The most common use of Oracle Net Services is to allow incoming database connections. You can configure additional net services to allow access to external code libraries (EXTPROC) and to connect the Oracle instance to non-Oracle data sources (such as Sybase, Informix, DB2, and SQL Server) through Oracle Heterogeneous Services.

Oracle Net Listener: Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Net Listener (or simply *the listener*) is the gateway to the Oracle instance for all nonlocal user connections. A single listener can service multiple database instances and thousands of client connections.

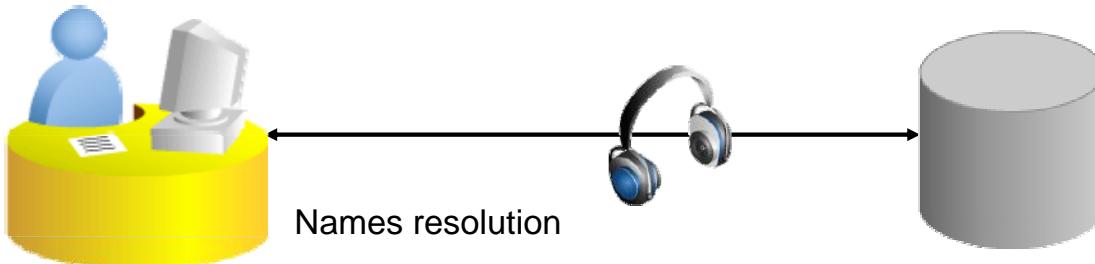
You can use Enterprise Manager Cloud Control or Oracle Net Manager to configure the listener and specify log file locations.

Advanced administrators can also configure Oracle Net Services by manually editing the configuration files, if necessary, with a standard operating system (OS) text editor such as vi or gedit.

Establishing Oracle Network Connections

To make a client or middle-tier connection, Oracle Net requires the client to know the:

- Host where the listener is running
- Port that the listener is monitoring
- Protocol that the listener is using
- Name of the service that the listener is handling



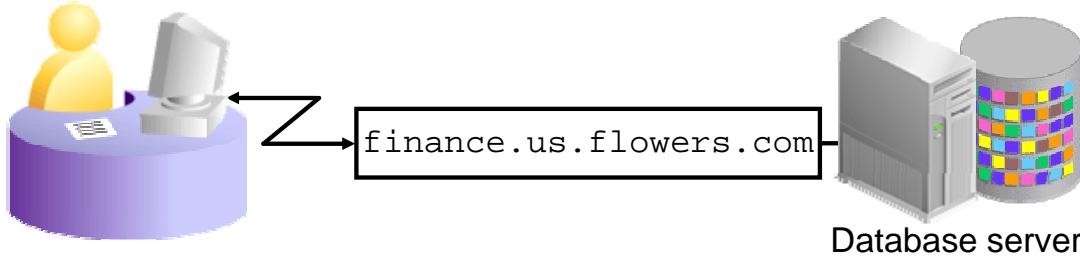
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For an application to connect to a service through Oracle Net Listener, the application must have information about that service, including the address or host where the listener resides, the protocol that the listener accepts, and the port that the listener monitors. After the listener is located, the final piece of information that the application needs is the name of the service to which it wants to connect.

Oracle Net names resolution is the process of determining this connection information.

Connecting to an Oracle Database



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An Oracle database is represented to a client as a service. A database can have one or more services associated with it. Databases are identified by a *service name* that is specified by the `SERVICE_NAMES` parameter in the initialization parameter file. The service name defaults to the global database name, which is a name that comprises the database name (`DB_NAME` parameter value) and the domain name (`DB_DOMAIN` parameter value).

To connect to a database service, clients use a *connect descriptor* that provides the location of the database and the name of the database service. Clients can use the connect descriptor or a name that resolves to the connect descriptor (as discussed later in this lesson).

The following example shows a connect descriptor that enables clients to connect to a database service called `finance.us.flowers.com`.

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=flowers-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=finance.us.flowers.com) ))
```

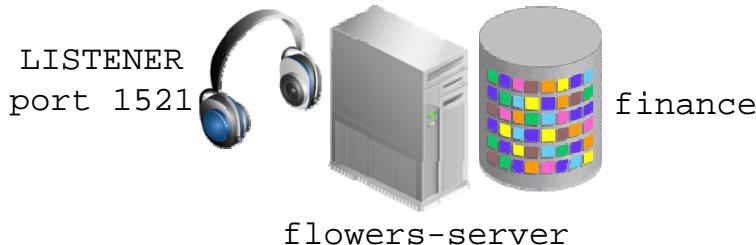
Name Resolution



```
CONNECT jsmith/jspass@finflowers
```

Name resolution

```
finflowers =(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=flowers-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=finance.us.flowers.com)))
```



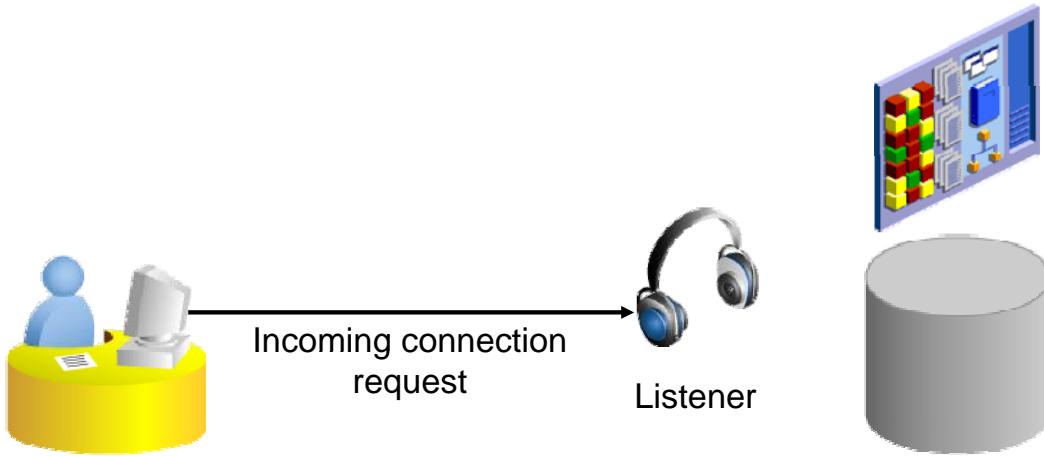
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Users initiate a connection request to the Oracle database by sending a *connect string*. A connect string includes a username and password, along with a *connect identifier*. A connect identifier can be the connect descriptor itself or a *name* that resolves to a connect descriptor. One of the most common connect identifiers is a *net service name*, which is a simple name for a service.

When a net service name is used, connection processing takes place by mapping the net service name to a connect descriptor. The mapping information can be stored in one or more repositories of information and is resolved using a *naming method*.

Establishing a Connection



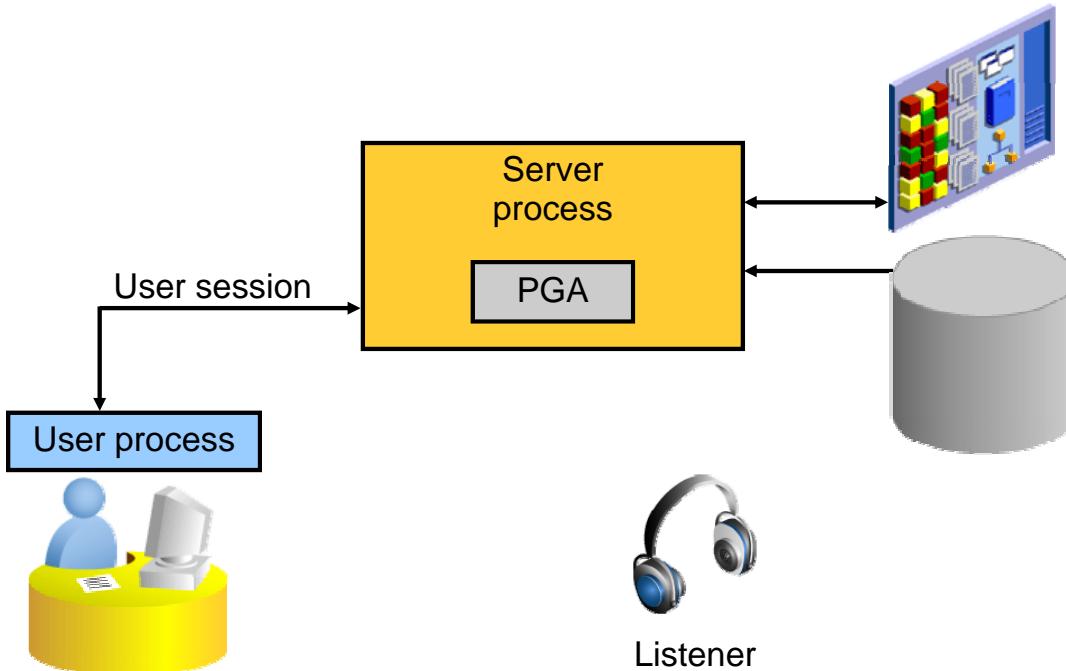
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After Oracle Net names resolution is complete, a connection request is passed from the user or middle-tier application (hereafter referred to as the *user process*) to the listener. The listener receives a CONNECT packet and checks whether that CONNECT packet is requesting a valid Oracle Net service name.

If the service name is not requested (as in the case of a `tnsping` request), the listener acknowledges the connect request and does nothing else. If an invalid service name is requested, the listener transmits an error code to the user process.

User Sessions



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If the CONNECT packet requests a valid service name, the listener spawns a new process to deal with the connection. This new process is known as the *server process*. The listener connects to the process and passes the initialization information, including the address information for the user process. At this point, the listener no longer deals with the connection and all work is passed to the server process.

The server process checks the user's authentication credentials (usually a password), and if the credentials are valid, a user session is created.

Dedicated server process: With the session established, the server process now acts as the user's agent on the server. The server process is responsible for:

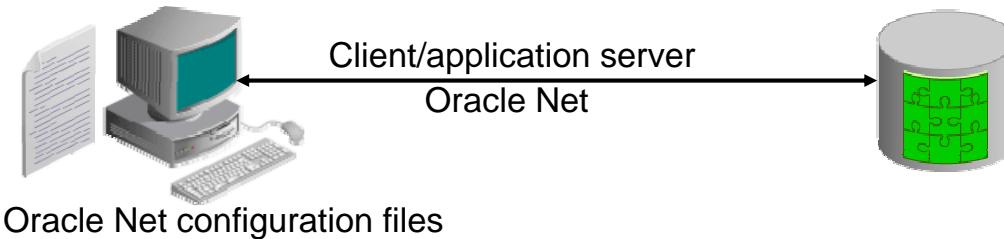
- Parsing and running any SQL statements issued through the application
- Checking the database buffer cache for data blocks required to perform SQL statements
- Reading necessary data blocks from data files on the disk into the database buffer cache portion of the System Global Area (SGA), if the blocks are not already present in the SGA
- Managing all sorting activity. The Sort Area is a memory area that is used to work with sorting; it is contained in a portion of memory that is associated with the Program Global Area (PGA).

- Returning results to the user process in such a way that the application can process the information
- Reading auditing options and reporting user processes to the audit destination

Naming Methods

Oracle Net supports several methods of resolving connection information:

- Easy connect naming: Uses a TCP/IP connect string
- Local naming: Uses a local configuration file
- Directory naming: Uses a centralized LDAP-compliant directory server
- External naming: Uses a supported non-Oracle naming service



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Net provides support for the following naming methods:

- **Easy connect naming:** The easy connect naming method enables clients to connect to an Oracle Database server by using a TCP/IP connect string consisting of a host name and optional port and service name as follows:

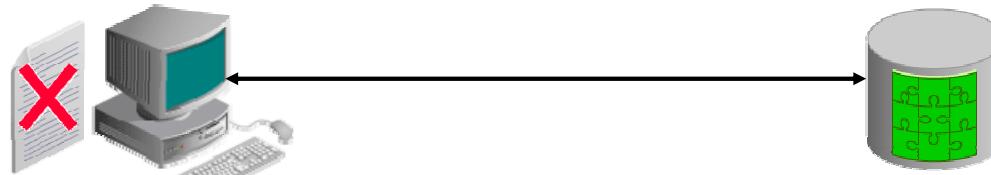
```
CONNECT username/password@host[:port][/:service_name]
```

The easy connect naming method requires no configuration.
- **Local naming:** The local naming method stores connect descriptors (identified by their net service name) in a local configuration file named `tnsnames.ora` on the client.
- **Directory naming:** To access a database service, the directory naming method stores connect identifiers in a centralized directory server that is compliant with the Lightweight Directory Access Protocol (LDAP).
- **External naming:** The external naming method stores net service names in a supported non-Oracle naming service. Supported third-party services include:
 - Network Information Service (NIS) External Naming
 - Distributed Computing Environment (DCE) Cell Directory Services (CDS)

Easy Connect

- Is enabled by default
- Requires no client-side configuration
- Supports only TCP/IP (no SSL)
- Offers no support for advanced connection options such as:
 - Connect-time failover
 - Source routing
 - Load balancing

```
SQL> CONNECT hr/hr@db.us.oracle.com:1521/dba11g
```



No Oracle Net configuration files

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

With Easy Connect, you supply all information that is required for the Oracle Net connection as part of the connect string. Easy Connect connection strings take the following form:

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

The listener port and service name are optional. If the listener port is not provided, Oracle Net assumes that the default port of 1521 is being used. If the service name is not provided, Oracle Net assumes that the database service name and host name provided in the connect string are identical.

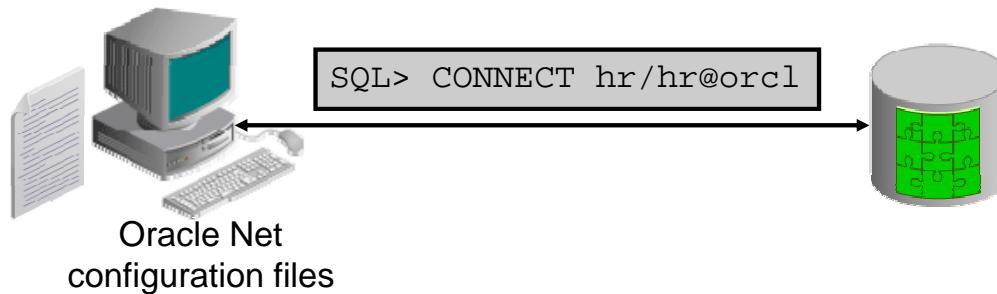
Assuming that the listener uses TCP to listen on port 1521 and the SERVICE_NAMES=db and DB_DOMAIN=us.oracle.com instance parameters, the connect string shown in the slide can be shortened:

```
SQL> connect hr/hr@db.us.oracle.com
```

Note: The SERVICE_NAMES initialization parameter can accept multiple comma-separated values. Only one of those values must be db for this scenario to work.

Local Naming

- Requires a client-side names-resolution file
- Supports all Oracle Net protocols
- Supports advanced connection options such as:
 - Connect-time failover
 - Source routing
 - Load balancing



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

With local naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against a local list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name.

One advantage of local naming is that the database users need to remember only a short alias rather than the long connect string required by Easy Connect.

The local list of known services is stored in the following text configuration file:

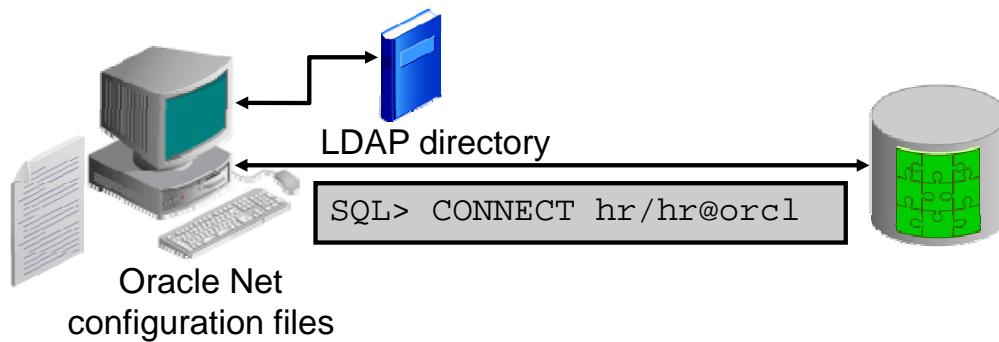
<oracle_home>/network/admin/tnsnames.ora

This is the default location of the `tnsnames.ora` file, but the file can be located elsewhere using the `TNS_ADMIN` environment variable.

Local naming is appropriate for organizations in which Oracle Net service configurations do not change often.

Directory Naming

- Requires LDAP with Oracle Net names resolution information loaded:
 - Oracle Internet Directory
 - Microsoft Active Directory Services
- Supports all Oracle Net protocols
- Supports advanced connection options



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

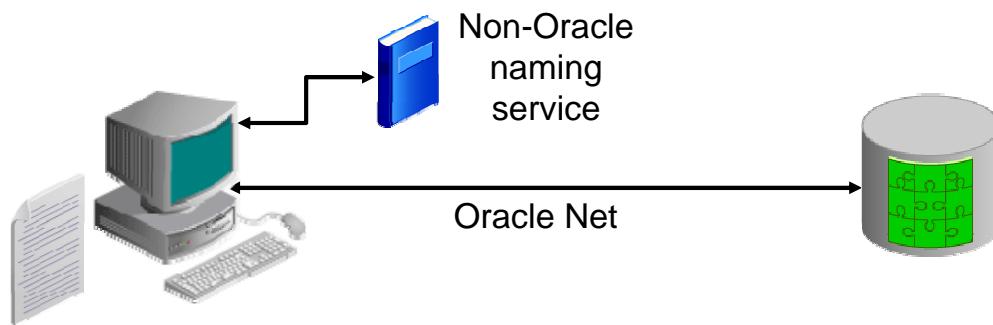
With directory naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against an external list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name. Like local naming, database users need to remember only a short alias.

One advantage of directory naming is that the service name is available for users to connect with as soon as a new service name is added to the LDAP directory. With local naming, the database administrator (DBA) must first distribute updated `tnsnames.ora` files containing the changed service name information before users can connect to new or modified services.

Directory naming is appropriate for organizations in which Oracle Net service configurations change frequently.

External Naming Method

- Uses a supported non-Oracle naming service
- Includes:
 - Network Information Service (NIS) External Naming
 - Distributed Computing Environment (DCE) Cell Directory Services (CDS)



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The external naming method stores Net Service names in a supported non-Oracle naming service. Supported third-party services include:

- Network Information Service (NIS) External Naming
- Distributed Computing Environment (DCE) Cell Directory Services (CDS)

Conceptually, external naming is similar to directory naming.

Tools for Configuring and Managing Oracle Net Services

- Enterprise Manager Net Services Administration page
- Oracle Net Manager
- Oracle Net Configuration Assistant
- Listener Control Utility



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Use the following tools and applications to manage your Oracle Network configuration:

- **Enterprise Manager Cloud Control:** Provides an integrated environment for configuring and managing Oracle Net Services. Use Enterprise Manager to configure Oracle Net Services for any Oracle home across multiple file systems and to administer listeners.
- **Oracle Net Manager:** Provides a graphical user interface (GUI) through which you can configure Oracle Net Services for an Oracle home on a local client or a server host.
- **Oracle Net Configuration Assistant:** Launched by Oracle Universal Installer when you install the Oracle software. During a typical database installation, Oracle Net Configuration Assistant automatically configures a listener called LISTENER that has a TCP/IP listening protocol address for the database. If you perform a custom installation, Oracle Net Configuration Assistant prompts you to configure a listener name and protocol address of your choice.
- **Listener Control Utility:** Used to start, stop, and view the status of the listener process

Defining Oracle Net Services Components

Component	Description	File
Listeners	A process that resides on the server whose responsibility is to listen for incoming client connection requests and manage the traffic to the server.	listener.ora
Naming methods	A resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service.	
Naming (net service name)	A simple name (connect identifier) for a service that resolves to a connect descriptor to identify the network location and identification of a service.	tnsnames.ora (local configuration)
Profiles	A collection of parameters that specifies preferences for enabling and configuring Oracle Net features on the client or server.	sqlnet.ora



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following Oracle Net Services components can be configured by using Enterprise Manager Cloud Control and Oracle Net Manager:

- Listener: Configuration of the listener includes specifying the listener name, protocol addresses it is accepting connection requests on, and services (database or non-database service) it is listening for.
- Naming (net service name)
- Naming methods
- Profiles

The Oracle Net Configuration Assistant configures the listener, naming methods, directory server usage, and a local `tnsnames.ora` file during installation of Oracle Database software.

Using Enterprise Manager Cloud Control

The screenshot shows two windows of the Oracle Enterprise Manager Cloud Control 12c interface. The top window is a navigation bar with links for Enterprise, Targets, Favorites, and History. Below it, a secondary window is open for the host 'edRSr9p1.us.oracle.com'. This window has a sidebar with options like Home, Monitoring, Control, Job Activity, Information Publisher Reports, Administration, and Net Services Administration. The 'Net Services Administration' link is highlighted with a yellow box and a red arrow points to it from the main navigation bar. The main content area of the second window displays the 'Net Services Administration' page, which lists components such as Listener, Directory Naming, Local Naming, Network Profile, File Location, and Group Copy Of Network Config Files. A sub-menu for 'Listeners' is open, showing options like Select, C, and Go, with 'Listeners' selected.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To administer Oracle Net Services in Enterprise Manager Cloud Control, select Net Services Administration in the Host menu for your target host.

The Net Services Administration page enables you to configure Oracle Net Services for any Oracle home across multiple file systems.

It also provides common administration functions for listeners such as starting and stopping a listener, and changing its tracing and logging characteristics. You can also look at a listener's control status report.

You can also specify the ORACLE_HOME location where the Oracle Net Services configuration files are stored by selecting "File Location."

Using Oracle Net Manager

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red background.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Use Oracle Net Manager to configure Oracle Net Services for an Oracle home on a local client or server host.

You can invoke Oracle Net Manager in the following ways:

- On Linux, enter `netmgr` at the operating system prompt.
- On Microsoft Windows select:
 - Programs from the Start menu
 - Oracle - *HOME_NAME*
 - Configuration and Migration Tools
 - Net Manager

Using Oracle Net Configuration Assistant



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Net Configuration Assistant is executed by the Oracle Universal Installer during installation of Oracle Database software. It configures the following basic network components:

- Listener names and protocol addresses
- Naming methods
- Net service names
- Directory server usage: Configures a directory server for directory-enabled features

Using the Listener Control Utility

```
$ lsnrctl
LSNRCTL for Linux: Version 12.1.0.2.0 - Production on 08-OCT-2014 10:07:23

Copyright (c) 1991, 2014, Oracle. All rights reserved.

Welcome to LSNRCTL, type "help" for information.

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

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Listener Control Utility enables you to control the listener. With `lsnrctl`, you can:

- Start the listener
- Stop the listener
- Check the status of the listener
- Reinitialize the listener from the configuration file parameters
- Dynamically configure many listeners
- Change the listener password

The basic command syntax for this utility is:

```
LSNRCTL> command [listener_name]
```

When the `lsnrctl` command is issued, the command acts on the default listener (named `LISTENER`) unless a different listener name is specified or the `SET CURRENT_LISTENER` command is executed. If the listener name is `LISTENER`, the `listener_name` argument can be omitted. The valid commands for `lsnrctl` are shown in the slide.

Note: The `lsnrctl` utility is located in both the Grid Infrastructure home and the Oracle Database home. It is important to set the environment variables to the appropriate home before using it.

Listener Control Utility Syntax

Commands for the Listener Control Utility can be issued from the command line or from the lsnrctl prompt.

- Command-line syntax:

```
$ lsnrctl <command name>
$ lsnrctl start
$ lsnrctl status
```

- Prompt syntax:

```
LSNRCTL> <command name>
LSNRCTL> start
LSNRCTL> status
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The lsnrctl commands can be issued from within the utility (prompt syntax) or from the command line. The following two commands have the same effect but use command-line syntax and prompt syntax, respectively:

Command-line syntax:

```
$ lsnrctl start
```

Prompt syntax:

```
$ lsnrctl
LSNRCTL for Linux: Version 12.1.0.2.0 - Production on 08-OCT-2014
10:07:23
Copyright (c) 1991, 2014, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> start
```

The command-line syntax is typically used to execute an individual command or scripted commands. If you plan to execute several consecutive lsnrctl commands, the prompt syntax is more efficient. Note that the `listener_name` argument is omitted, and the command would thus affect the listener named LISTENER.

Remember that if your listener is named something other than LISTENER, you must either include the listener name with the command or use the SET CURRENT_LISTENER command. Suppose that your listener is named custom_lis. Here are two examples of stopping a listener named custom_lis by using prompt syntax:

```
LSNRCTL> stop custom_lis
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)(PORT=5521)))
The command completed successfully
```

This produces the same results as the following:

```
LSNRCTL> set cur custom_lis
Current Listener is custom_lis
LSNRCTL> stop
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)(PORT=5521)))
The command completed successfully
```

Note: In the preceding syntax, current_listener has been abbreviated to cur.

Using command-line syntax produces the same results:

```
$ lsnrctl stop custom_lis
LSNRCTL for Linux: Version 12.1.0.2.0 - Production on 08-OCT-
2014 10:07:23
Copyright (c) 1991, 2014, Oracle. All rights reserved.
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)(PORT=5521)))
The command completed successfully
```

Advanced Connection Options

Oracle Net supports the following advanced connection options with local and directory naming:

- Connect-time failover
- Load balancing
- Source routing



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When a database service is accessible by multiple listener protocol addresses, you can specify the order in which the addresses are to be used. The addresses can be chosen randomly or tried sequentially. In cases in which more than one listener is available, such as Oracle Real Application Clusters (RAC) configurations, Oracle Net can take advantage of listener failover and load balancing as well as Oracle Connection Manager source routing.

With *connect-time failover* enabled, the alias has two or more listener addresses listed. If the first address is not available, the second is tried. Oracle Net keeps trying addresses in the listed order until it reaches a listener that is functioning or until all addresses have been tried and failed. Transparent Application Failover (TAF) is a client-side feature that allows clients to reconnect to surviving databases in the event of a database instance failure. Notifications are used by the server to trigger TAF callbacks on the client side.

With *load balancing* enabled, Oracle Net picks an address at random from the list of addresses. The runtime connection load-balancing feature improves connection performance by balancing the number of active connections among multiple dispatchers. In a RAC environment, connection pool load balancing also has the capability to balance the number of active connections among multiple instances.

Source routing is used with Oracle Connection Manager, which serves as a proxy server for Oracle Net traffic, enabling Oracle Net traffic to be routed securely through a firewall. Oracle Net treats the addresses as a list of relays, connecting to the first address and then requesting to be passed from the first to the second until the destination is reached. It differs from failover or load balancing in that all addresses are used each time a connection is made.

Testing Oracle Net Connectivity

The `tnsping` utility that tests Oracle Net service aliases:

- Ensures connectivity between the client and the Oracle Net Listener
- Does not verify that the requested service is available
- Supports Easy Connect Names Resolution:
`tnsping host01.example.com:1521/orcl`
- Supports local and directory naming:
`tnsping orcl`



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

`tnsping` is the Oracle Net equivalent of the TCP/IP ping utility. It offers a quick test to verify that the network path to a destination is good. For example, enter `tnsping orcl` in a command-line window.

The utility validates that the host name, port, and protocol reach a listener. It does not actually check whether the listener handles the service name. The `tnsping` utility also reveals the location of the configuration files. In a system with multiple `ORACLE_HOME` locations, this can be helpful.

Comparing Dedicated Server and Shared Server Configurations

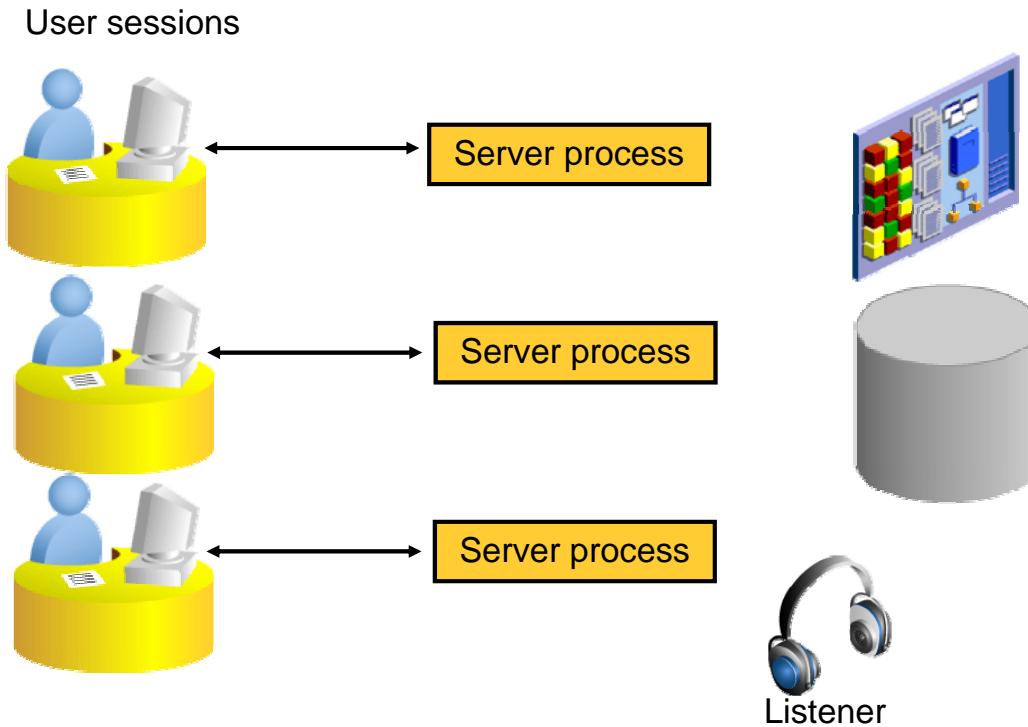
- Dedicated server configuration: One server process for each client
- Shared server configuration: A small pool of server processes can serve a large number of clients



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In a dedicated server configuration a server process handles requests for a single client process. A shared server configuration enables multiple client processes to share a small number of server processes. Detailed information on each configuration follows in this lesson.

User Sessions: Dedicated Server Process



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

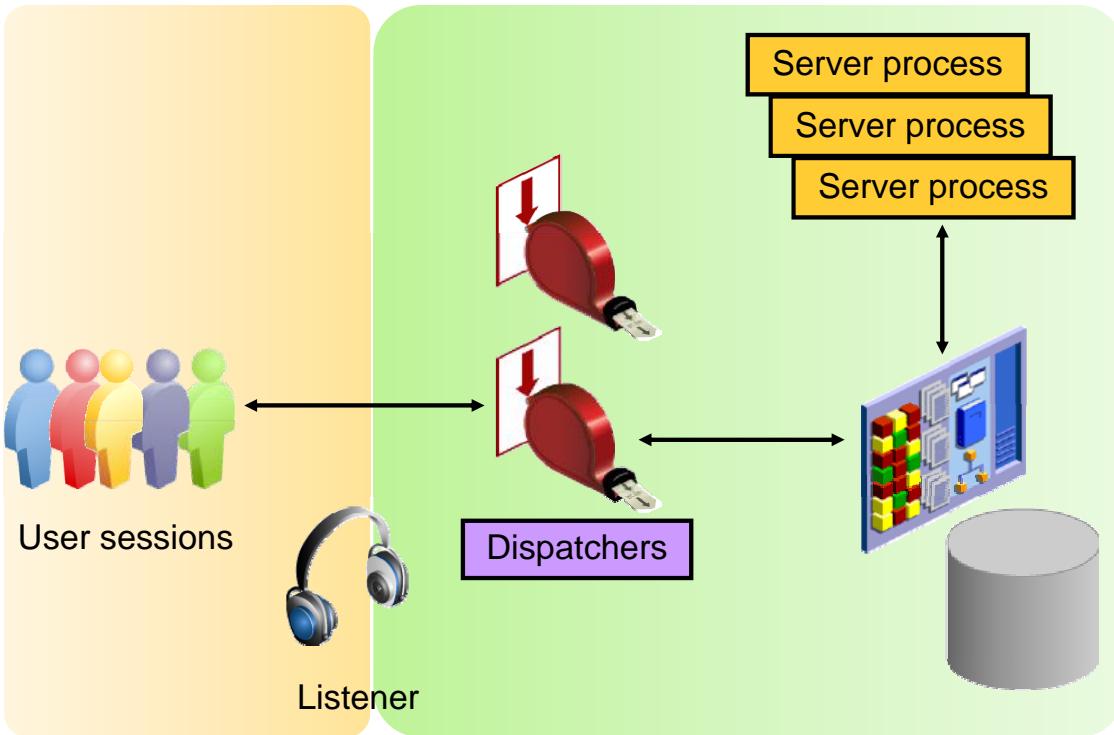
ORACLE

With dedicated server processes, there is a one-to-one ratio of server processes to user processes. Each server process uses system resources, including CPU cycles and memory.

In a heavily loaded system, the memory and CPU resources that are used by dedicated server processes can be prohibitive and can negatively affect the system's scalability. If your system is being negatively affected by the resource demands of the dedicated server architecture, you have the following options:

- Increasing system resources by adding more memory and additional CPU capability
- Using the Oracle Shared Server Process architecture

User Sessions: Shared Server Processes



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each service that participates in the shared server process architecture has at least one dispatcher process (and usually more). When a connection request arrives, the listener does not spawn a dedicated server process. Instead, the listener maintains a list of dispatchers that are available for each service name, along with the connection load (number of concurrent connections) for each dispatcher.

Connection requests are routed to the lightest loaded dispatcher that is servicing a given service name. Users remain connected to the same dispatcher for the duration of a session.

Unlike dedicated server processes, a single dispatcher can manage hundreds of user sessions.

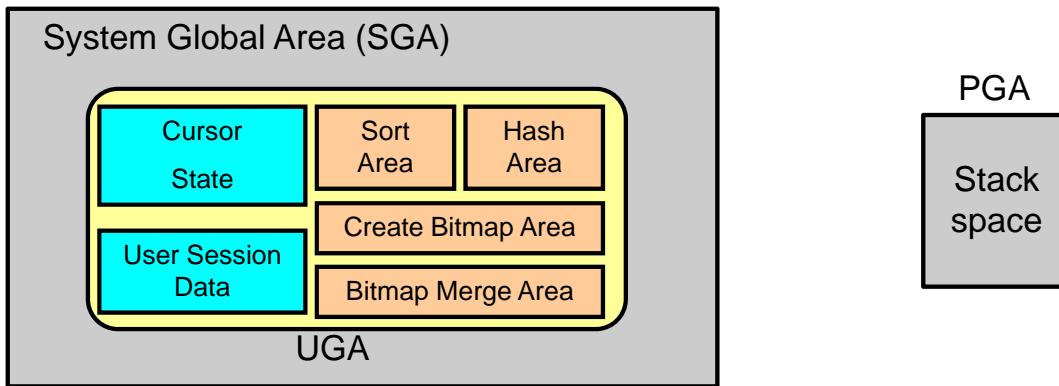
Dispatchers do not actually handle the work of user requests. Instead, they pass user requests to a common queue located in the shared pool portion of the SGA.

Shared server processes take over most of the work of dedicated server processes, pulling requests from the queue and processing them until they are complete.

Because a user session may have requests processed by multiple shared server processes, most of the memory structures that are usually stored in the PGA must be in a shared memory location (by default, in the shared pool). However, if the large pool is configured or if SGA_TARGET is set for automatic memory management, these memory structures are stored in the large pool portion of the SGA.

SGA and PGA Usage

Oracle Shared Server: User session data is held in the SGA.



Remember to consider shared server memory requirements when sizing the SGA.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The contents of the SGA and PGA differ when dedicated servers or shared servers are used:

- Text and parsed forms of all SQL statements are stored in the SGA.
- The cursor state contains runtime memory values for the SQL statement, such as rows retrieved.
- User-session data includes security and resource usage information.
- The stack space contains local variables for the process.

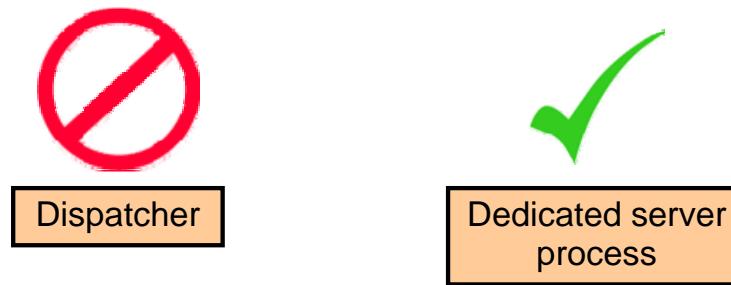
Technical Note

The change in the SGA and PGA is transparent to the user; however, if you are supporting multiple users, you need to increase the `LARGE_POOL_SIZE` initialization parameter. Each shared server process must access the data spaces of all sessions so that any server can handle requests from any session. Space is allocated in the SGA for each session's data space. You limit the amount of space that a session can allocate by setting the `PRIVATE_SGA` resource.

Shared Server Configuration Considerations

Certain types of database work must not be performed using shared servers:

- Database administration
- Backup and recovery operations
- Batch processing and bulk load operations
- Data warehouse operations



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Oracle Shared Server architecture is an efficient process and memory use model, but it is not appropriate for all connections. Because of the common request queue and the fact that many users may share a dispatcher response queue, shared servers do not perform well with operations that must deal with large sets of data, such as warehouse queries or batch processing.

Backup and recovery sessions that use Oracle Recovery Manager (discussed in later lessons) also deal with very large data sets and must use dedicated connections.

Many administration tasks must not (and cannot) be performed by using shared server connections. These include starting up and shutting down the instance, creating tablespaces and data files, maintaining indexes and tables, analyzing statistics, and many other tasks that are commonly performed by the DBA. All DBA sessions must choose dedicated servers.

Configuring Communication Between Databases

- Sending data or messages between sites requires network configuration on both sites.
- You must configure the following:
 - Network connectivity (for example, `tnsnames.ora`)
 - Database links

```
CREATE DATABASE LINK <remote_global_name>
CONNECT TO <user> IDENTIFIED BY <pwd>
USING '<connect_string_for_remote_db>' ;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A database link is a schema object in one database that enables you to access objects on another database. The other database need not be an Oracle database system. However, to access non-Oracle systems, you must use Oracle Heterogeneous Services.

To create a private database link, you must have the `CREATE DATABASE LINK` system privilege. To create a public database link, you must have the `CREATE PUBLIC DATABASE LINK` system privilege.

When an application uses a database link to access a remote database, Oracle Database establishes a database session in the remote database on behalf of the local request. The `CONNECT TO` clause that is used in creating a database link determines how the connection is established on the remote database. You can create fixed user, current user, and connected user database links. Current user links are available only through the Oracle Advanced Security option. The example in the slide shows the syntax to create a fixed user database link.

After you create a database link, you can use it to refer to tables and views on the other database. In SQL statements, you can refer to a table or view on the other database by appending `@dblink` to the table or view name. You can query a table or view on the other database or use any `INSERT`, `UPDATE`, `DELETE`, or `LOCK TABLE` statement for the table.

Connecting to Another Database

```
REMOTE_ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)
    (HOST = host02.example.com)
    (PORT = 1521))
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = orcl2.example.com)
  )
)
```

tnsnames.ora

```
CONNECT hr/hr@orcl1;
```

SQL*Plus

```
CREATE DATABASE LINK remote
CONNECT TO HR IDENTIFIED BY HR
USING 'REMOTE_ORCL' ;
```

```
SELECT * FROM employees@remote
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The slide shows the `tnsnames.ora` entry that is needed before creating a database link. The example shows a fixed user database link called `REMOTE` that is connecting to the user `HR` by using the connect string `REMOTE_ORCL`. After you create a database link, you can use it to refer to tables and views on the other database.

The description of the view is as follows:

SQL> DESC DBA_DB_LINKS		
Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
DB_LINK	NOT NULL	VARCHAR2(128)
USERNAME		VARCHAR2(30)
HOST		VARCHAR2(2000)
CREATED	NOT NULL	DATE

SQL> select owner, db_link, username from dba_db_links;		
OWNER	DB_LINK	USERNAME
HR	REMOTE.EXAMPLE.COM	HR

Quiz

Which configuration files are used to configure the listener?

- a. listener.ora
- b. listener.conf
- c. tnsnames.ora
- d. tnsnames.conf
- e. sqlnet.ora
- f. sqlnet.conf



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, e

Quiz

When using the shared server process architecture, the UGA portion of the PGA is relocated into the SGA.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager to:
 - Create additional listeners
 - Create Oracle Net Service aliases
 - Control the Oracle Net Listener
- Use `tnsping` to test Oracle Net connectivity
- Identify when to use shared servers and when to use dedicated servers



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Configuring local names resolution to connect to another database
- Creating a second listener



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Oracle University and Error : You are not a Valid Partner use only

Administering User Security



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Create and manage database user accounts:
 - Authenticate users
 - Assign default storage areas (tablespaces)
- Grant and revoke privileges
- Create and manage roles
- Create and manage profiles:
 - Implement standard password security features
 - Control resource usage by users



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

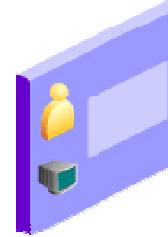
The following terms relate to administering database users and assist you in understanding the objectives:

- A *database user account* is a way to organize the ownership of and access to database objects.
- A *password* is an authentication by the Oracle database.
- A *privilege* is a right to execute a particular type of SQL statement or to access another user's object.
- A *role* is a named group of related privileges that are granted to users or to other roles.
- *Profiles* impose a named set of resource limits on database usage and instance resources, and manage account status and password management rules.
- A *quota* is a space allowance in a given tablespace. This is one of the ways by which you can control resource usage by users.

Database User Accounts

Each database user account has:

- A unique username
- An authentication method
- A default tablespace
- A temporary tablespace
- A user profile
- An initial consumer group
- An account status



A schema:

- Is a collection of database objects that are owned by a database user
- Has the same name as the user account

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To access the database, a user must specify a valid database user account and successfully authenticate as required by that user account. Each database user has a unique database account.

Oracle recommends this to avoid potential security holes and provide meaningful data for certain audit activities. However, users may sometimes share a common database account. In these rare cases, the operating system and applications must provide adequate security for the database. Each user account has:

- **Unique username:** Usernames cannot exceed 30 bytes, cannot contain special characters, and must start with a letter.
- **Authentication method:** The most common authentication method is a password. Oracle Database supports password, global, and external authentication methods (such as biometric, certificate, and token authentication).
- **Default tablespace:** This is a place where a user creates objects if the user does not specify some other tablespace. Note that having a default tablespace does not imply that the user has the *privilege* of creating objects in that tablespace, nor does the user have a *quota* of space in that tablespace in which to create objects. Both of these are granted separately.

- **Temporary tablespace:** This is a place where temporary objects, such as sorts and temporary tables, are created on behalf of the user by the instance. No quota is applied to temporary tablespaces.
- **User profile:** This is a set of resource and password restrictions assigned to the user.
- **Initial consumer group:** This is used by the Resource Manager.
- **Account status:** Users can access only “open” accounts. The account status may be “locked” and/or “expired.”

Schemas: A schema is a collection of database objects that are owned by a database user. Schema objects are the logical structures that directly refer to the database’s data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, schema objects include everything that your application creates in the database.

Note: A database user is not necessarily a person. It is a common practice to create a user that owns the database objects of a particular application, such as HR. The database user can be a device, an application, or just a way to group database objects for security purposes. The personal identifying information of a person is not needed for a database user.

Predefined Administrative Accounts

- SYS:
 - Owns the data dictionary and the Automatic Workload Repository (AWR)
 - Used for startup and shutdown of the database instance
- SYSTEM: Owns additional administrative tables and views
- SYSBACKUP: Facilitates Oracle Recovery Manager (RMAN) backup and recovery operations
- SYSDG: Facilitates Oracle Data Guard operations
- SYSKM: Facilitates Transparent Data Encryption wallet operations



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The SYS and SYSTEM accounts have the database administrator (DBA) role granted to them by default. In addition, the SYS account has all privileges with ADMIN OPTION and owns the data dictionary. To connect to the SYS account, you must use the AS SYSDBA clause for a database instance and AS SYSASM for an Automatic Storage Management (ASM) instance. Any user that is granted the SYSDBA privilege can connect to the SYS account by using the AS SYSDBA clause. Only “privileged” users who are granted the SYSDBA, SYSOPER, or SYSASM privileges are allowed to start up and shut down instances. The SYSTEM account does not have the SYSDBA privilege. SYSTEM is also granted the AQ_ADMINISTRATOR_ROLE and MGMT_USER roles. The SYS and SYSTEM accounts are required accounts in the database. They cannot be dropped.

Best practice: Applying the principle of least privilege, these accounts are not used for routine operations. Users who need DBA privileges have separate accounts with the required privileges granted to them.

The SYSBACKUP, SYSDG, and SYSKM users are created to facilitate separation of duties for database administrators. Each of these provides a designated use for an administrative privilege by the same name. You should create a user and grant the appropriate administrative privilege to that user.

Privileges are discussed in detail later in the lesson.

Administrative Privileges

Privilege	Description
SYSDBA	Standard database operations, such as starting and shutting down the database instance, creating the server parameter file (SPFILE), and changing the ARCHIVELOG mode Allows the grantee to view user data
SYSOPER	Standard database operations, such as starting and shutting down the database instance, creating the server parameter file (SPFILE), and changing the ARCHIVELOG mode
SYSBACKUP	Oracle Recovery Manager (RMAN) backup and recovery operations by using RMAN or SQL*Plus
SYSDG	Data Guard operations by using the Data Guard Broker or the DGMGR command-line interface
SYSKM	Manage Transparent Data Encryption wallet operations



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

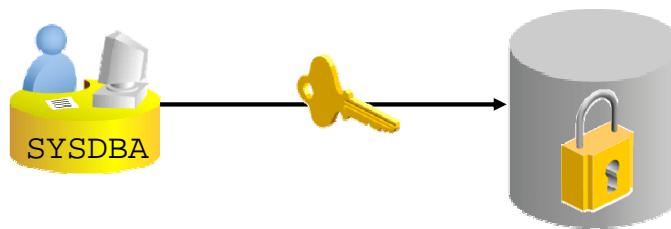
Oracle Database includes five administrative privileges that are provided to facilitate separation of duty. The SYSDBA and SYSOPER administrative privileges are used to perform a variety of standard database operations including starting up the database instance and shutting it down. Refer to the *Oracle Database Administrator's Guide* for a complete list of authorized operations for the SYSDBA and SYSOPER privileges.

SYSBACKUP, SYSDG, and SYSKM are new to Oracle Database 12c and are tailored for the specific administrative tasks of backup and recovery, Oracle Data Guard, and Transparent Data Encryption key management. In previous releases, the SYSDBA privilege was required for these tasks. These privileges enable you to connect to the database even if the database is not open. Refer to the *Oracle Database Security Guide* for a list of supported operations for the SYSBACKUP, SYSDG, and SYSKM privileges.

Protecting Privileged Accounts

Privileged accounts can be protected by:

- Using a password file with case-sensitive passwords
- Enabling strong authentication for administrator roles



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Users with SYSDBA or SYSOPER privileges must always be authenticated. When connecting locally, the user is authenticated by the local OS by being a member of a privileged OS group. If connecting remotely, a password file is used to authenticate privileged users. If the password file is configured, it will be checked first. In Oracle Database, these passwords are case-sensitive. Oracle Database provides other methods that make remote administrator authentication more secure and centralize the administration of these privileged users.

When a database is created using the Database Configuration Assistant, the password file is case-sensitive. If you upgrade from earlier database versions, be sure to make the password file case-sensitive for remote connections:

```
orapwd file=orapworcl entries=5 ignorecase=N
```

If your concern is that the password file might be vulnerable or that the maintenance of many password files is a burden, strong authentication can be implemented. The Advanced Security option is required if you want to use strong authentication methods. For more information about strong authentication, see the *Oracle Database Advanced Security Administrator's Guide*.

Authenticating Users

- Password: User definition includes a password that must be supplied when the user attempts to log in to the database
- External: Authentication by a method outside the database (operating system, Kerberos, or Radius)
- Global: Users are identified by using an LDAP-based directory service



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Authentication means verifying the identity of someone or something (a user, device, or other entity) that wants to use data, resources, or applications. Validating that identity establishes a trust relationship for further interactions. Authentication also enables accountability by making it possible to link access and actions to specific identities. After authentication, authorization processes can allow or limit the levels of access and action that are permitted to that entity.

When you create a user, you must decide on the authentication technique to use, which can be modified later.

Password: This is also referred to as authentication by the Oracle Database server. Create each user with an associated password that must be supplied when the user attempts to establish a connection. When setting up a password, you can expire the password immediately, which forces the user to change the password after first logging in. If you decide on expiring user passwords, make sure that users have the ability to change the password. Some applications do not have this functionality. All passwords created in Oracle Database are case-sensitive by default. These passwords may also contain multibyte characters and are limited to 30 bytes.

Passwords are always automatically and transparently encrypted by using the Advanced Encryption Standard (AES) algorithm during network (client/server and server/server) connections before sending them across the network.

External: This is authentication by a method outside the database (operating system, Kerberos, or Radius). The Advanced Security Option is required for Kerberos or Radius. Users can connect to the Oracle database without specifying a username or password. The Advanced Security Option (which is a strong authentication) allows users to be identified by using biometrics, x509 certificates, and token devices. With external authentication, your database relies on the underlying operating system, network authentication service, or external authentication service to restrict access to database accounts. A database password is not used for this type of login. If your operating system or network service permits, you can have it authenticate users. If you use operating system authentication, set the `OS_AUTHENT_PREFIX` initialization parameter and use this prefix in Oracle usernames. The `OS_AUTHENT_PREFIX` parameter defines a prefix that the Oracle database adds to the beginning of each user's operating system account name. The default value of this parameter is `OPS$` for backward compatibility with the previous versions of the Oracle software. The Oracle database compares the prefixed username with the Oracle usernames in the database when a user attempts to connect. For example, suppose that `OS_AUTHENT_PREFIX` is set as follows:

```
OS_AUTHENT_PREFIX=OPS$
```

If a user with an operating system account named `tsmith` needs to connect to an Oracle database and be authenticated by the operating system, the Oracle database checks whether there is a corresponding database user `OPS$tsmith` and, if so, allows the user to connect. All references to a user who is authenticated by the operating system must include the prefix, as seen in `OPS$tsmith`.

Note: The text of the `OS_AUTHENT_PREFIX` initialization parameter is case-sensitive on some operating systems. See the Oracle documentation that is specific to your operating system for more information about this initialization parameter.

Global: With the Oracle Advanced Security option, global authentication enables users to be identified by using an LDAP-based directory service.

For more information about advanced authentication methods, see the *Oracle Database Security* course.

Administrator Authentication

Operating system security:

- DBAs must have the OS privileges to create and delete files.
- Typical database users should not have the OS privileges to create or delete database files.

Administrator security:

- For SYSDBA and SYSOPER connections:
 - DBA user by name is audited for password file and strong authentication methods.
 - OS account name is audited for OS authentication.
 - OS authentication takes precedence over password file authentication for privileged users.
 - Password file uses case-sensitive passwords.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Operating system security: In UNIX and Linux, DBAs by default belong to the `oinstall` OS group, which has the required privileges to create and delete database files.

Administrator security: Connections for the privileged users SYSDBA, SYSOPER, and SYSASM are authorized only after verification with the password file or with the OS privileges and permissions. If OS authentication is used, the database does *not* use the supplied username and password. OS authentication is used if there is no password file, if the supplied username or password is not in that file, or if no username and password are supplied. The password file in Oracle Database 12c uses case-sensitive passwords by default.

However, if authentication succeeds by means of the password file, the connection is logged with the username. If authentication succeeds by means of the operating system, it is a CONNECT / connection that does not record the specific user.

Note: If you are a member of the OSDBA or OSOPER group for the operating system and you connect as SYSDBA or SYSOPER, you will be connected with the associated administrative privileges regardless of the username and password that you specify.

In Oracle Database 12c, a privileged user may use strong authentication methods: Kerberos, SSL, or directory authentication if the Advanced Security Option is licensed.

OS Authentication and OS Groups

Privileged Operating System Groups

SYS privileges are required to create a database using operating system (OS) authentication. Membership in OS Groups grants the corresponding SYS privilege, eg. membership in OSDBA grants the SYSDBA privilege.

Database Administrator (OSDBA) Group: dba

Database Operator (OSOPER) Group (Optional): opdba

Database Backup and Recovery (OSBACKUPDBA) Group: bkpdba

Data Guard Administrative (OSDGDBA) Group: dgdba

Encryption Key Management Administrative (OSKMDBA) Group: kmdba

Grant SYSBACKUP privileges to a group (OSBACKUPDBA). SYSBACKUP privileges are granted to members of the OSBACKUPDBA group.

Grant SYSKM privileges to a group (OSKMDBA). SYSKM privileges are granted to members of the OSKMDBA group.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In OS authentication, the groups are created and assigned specific names as part of the database installation process, provided that OS UNIX or WINDOWS groups are created and accounts are assigned to the appropriate operating-system-defined groups.

Membership in a UNIX or WINDOWS group affects your connection to the database as follows:

- If you are a member of the OSBACKUP group, and you specify AS SYSBACKUP when you connect to the database, you connect to the database with the SYSBACKUP administrative privilege under the SYSBACKUP user.
- If you are a member of the OSDG group, and you specify AS SYSDG when you connect to the database, you connect to the database with the SYSDG administrative privilege under the SYSDG user.
- If you are a member of the OSKM group, and you specify AS SYSKM when you connect to the database, you connect to the database with the SYSKM administrative privilege under the SYSKM user.

If they are not members of these OS groups, users will not be able to connect as administrative users with 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 in 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 in 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.

Managing Users

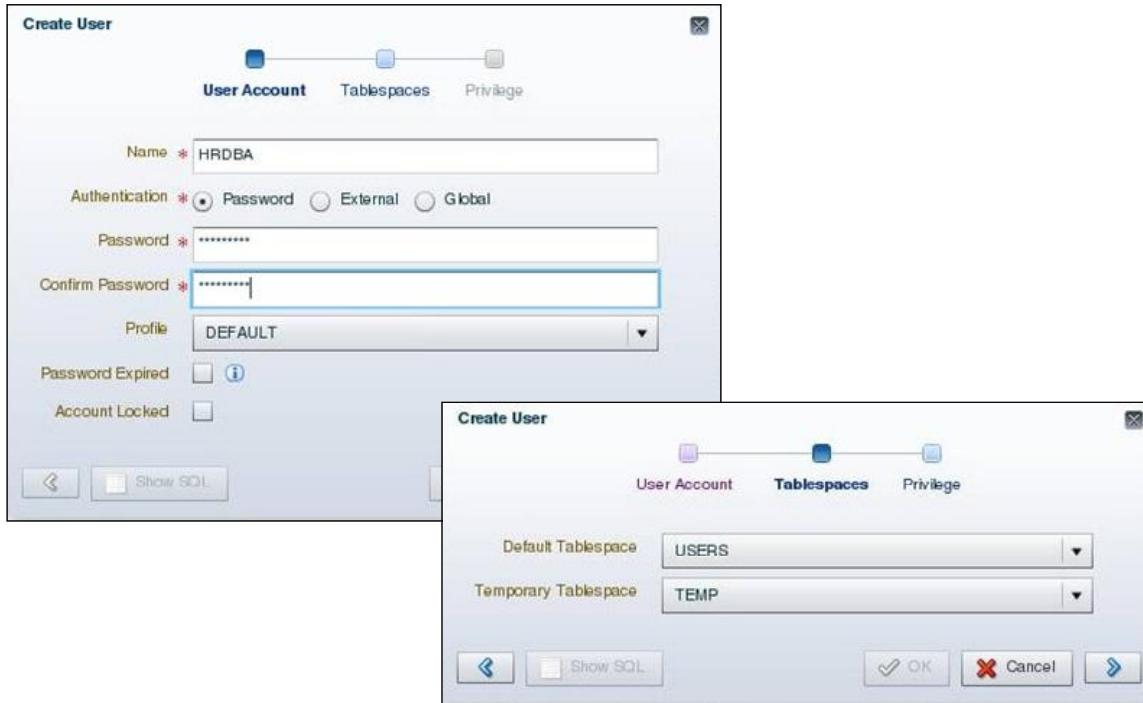
Name	Account Status	Expiration Date	Default Tablespace	Temporary Tablespace	Profile	Created
CTXSYS	locked	Mon Jul 7, 2014 6:22:03 /	SYSAUX	TEMP	DEFAULT	Mon Jul 7, 2014 6:11:11 AM
DBA1	checked	Sun Apr 5, 2015 10:37:11 /	SYSTEM	TEMP	DEFAULT	Tue Oct 7, 2014 10:37:18 AM
DBSNMP	checked	Mon Apr 6, 2015 11:58:11 /	SYSAUX	TEMP	DEFAULT	Mon Jul 7, 2014 5:53:06 AM
DIF	locked	Mon Jul 7, 2014 5:43:05 /	USERS	TEMP	DEFAULT	Mon Jul 7, 2014 5:43:05 AM
DUF	locked	Mon Jul 7, 2014 6:52:34 /	SYSAUX	TEMP	DEFAULT	Mon Jul 7, 2014 6:51:55 AM
DVSYS	locked	Mon Jul 7, 2014 6:52:07 /	SYSAUX	TEMP	DEFAULT	Mon Jul 7, 2014 6:51:55 AM
FLWS_FILES	locked	Mon Jul 7, 2014 6:31:11 /	SYSAUX	TEMP	DEFAULT	Mon Jul 7, 2014 6:29:17 AM
GSMADMIN_INTERNAL	locked	Mon Jul 7, 2014 5:42:58 /	SYSAUX	TEMP	DEFAULT	Mon Jul 7, 2014 5:42:58 AM
GSMCATUSER	locked	Mon Jul 7, 2014 5:56:36 /	USERS	TEMP	DEFAULT	Mon Jul 7, 2014 5:56:36 AM
GSMUSER	locked	Mon Jul 7, 2014 5:42:58 /	USERS	TEMP	DEFAULT	Mon Jul 7, 2014 5:42:58 AM
HR	checked	Mon Apr 6, 2015 11:58:11 /	USERS	TEMP	DEFAULT	Thu Oct 2, 2014 5:05:59 AM

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

On the Users page of Enterprise Manager Database Express, you manage the database users who are allowed to access the current database. You use this page to create, delete, and modify the settings of a user.

To access the Users page, expand the Security menu and select Users.

Creating a User



ORACLE

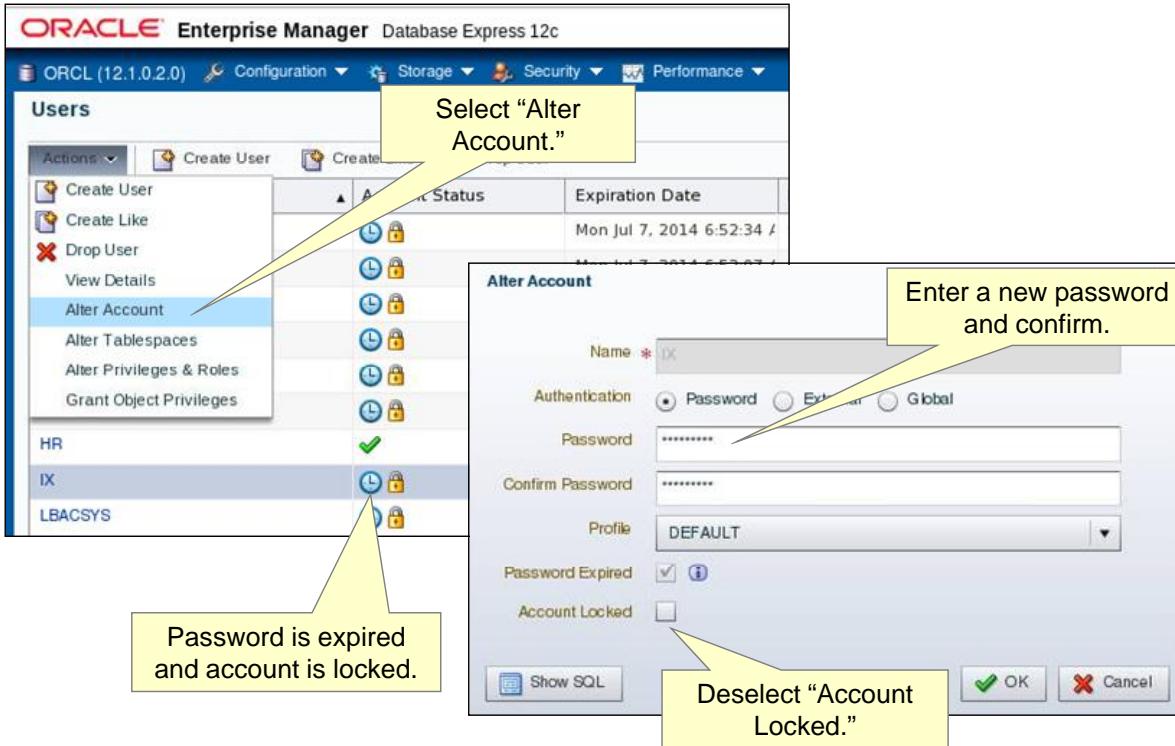
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a database user by using Enterprise Manager Database Express, select Create User on the Users page.

Provide the required information. Mandatory items (such as Name) are marked with an asterisk (*). The name specified must follow the same rules as those used for creating database objects. The pages that follow in this lesson give you more information about authentication. Profiles are covered later in this lesson.

Assign a default tablespace and a temporary tablespace to each user. If users do not specify a tablespace when creating an object, the object will be created in the default tablespace assigned to the object owner. This enables you to control where their objects are created. If you do not choose a default tablespace, the system-defined default permanent tablespace is used. The case is similar for the temporary tablespace: if you do not specify a tablespace, the system-defined temporary tablespace is used.

Unlocking a User Account and Resetting the Password



ORACLE

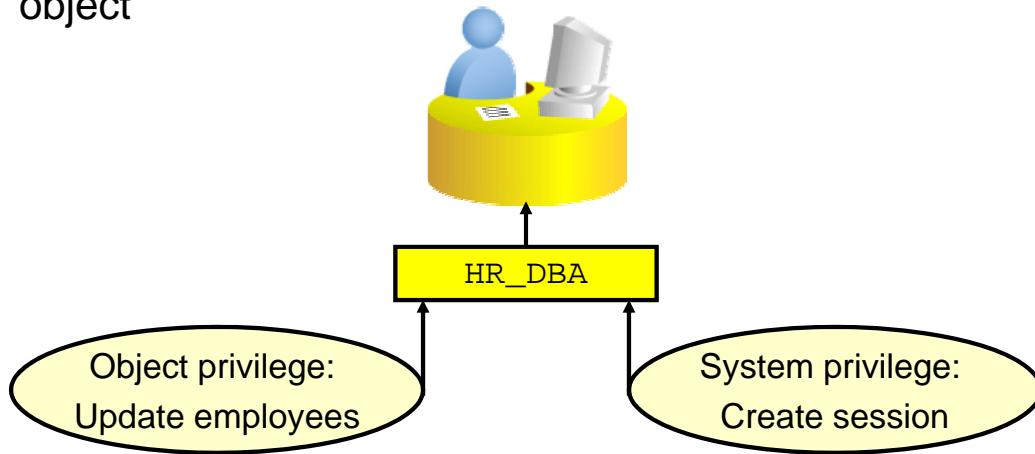
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

During installation and database creation, you can unlock and reset many of the Oracle-supplied database user accounts. If you did not choose to unlock the user accounts at that time, you can unlock the user by selecting the user on the Users page, selecting Alter User from the Actions list, and deselecting Account Locked. If the password is expired, enter a new password in the Password and Confirm Password fields.

Privileges

There are two types of user privileges:

- System: Enables users to perform particular actions in the database
- Object: Enables users to access and manipulate a specific object



ORACLE

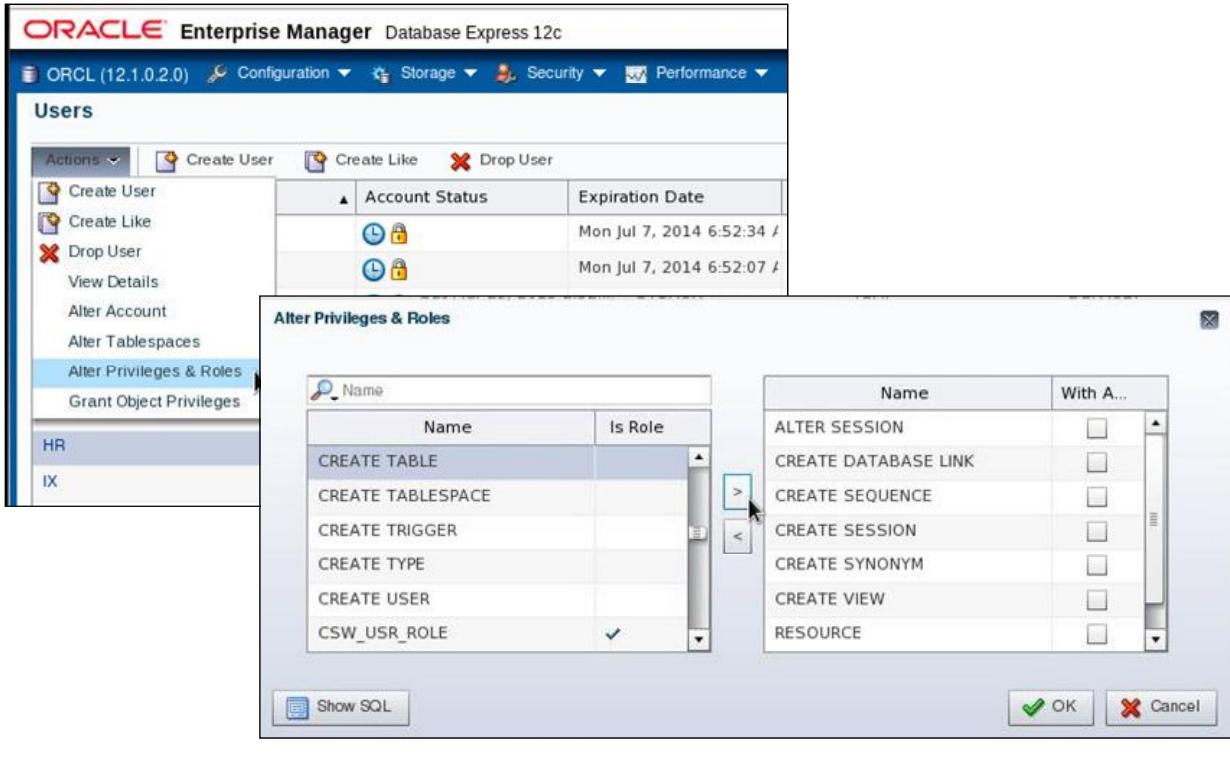
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A *privilege* is a right to execute a particular type of SQL statement or to access another user's object.

Privileges are divided into two categories:

- **System privileges:** Each system privilege allows a user to perform a particular database operation or class of database operations. For example, the privilege to create tablespaces is a system privilege. System privileges can be granted by the administrator or by someone who has been given explicit permission to administer the privilege. There are more than 170 distinct system privileges. Many system privileges contain the ANY clause.
- **Object privileges:** Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package. Without specific permission, users can access only their own objects. Object privileges can be granted by the owner of an object, by the administrator, or by someone who has been explicitly given permission to grant privileges on the object.

System Privileges



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can administer system privileges when you create a user or at a later time.

To grant or revoke system privileges, select the user, and then select “Alter Privileges & Roles” on the Users page. Select the appropriate privileges from the list of privileges and move them by clicking the appropriate arrow.

Granting a privilege with the ANY clause means that the privilege crosses schema lines. For example, if you have the CREATE TABLE privilege, you can create a table—but only in your own schema. The SELECT ANY TABLE privilege allows you to select from tables owned by other users. The SYS user and users with the DBA role are granted all of the ANY privileges; they can therefore do anything to any data object. The scope of the ANY system privileges can be controlled by using the Oracle Database Vault Option.

Select the With Admin check box to enable the user to administer the privilege and grant the system privilege to other users.

The SQL syntax for granting system privileges is:

```
GRANT <system_privilege> TO <grantee_clause> [WITH ADMIN OPTION]
```

Carefully consider security requirements before granting system permissions. Some system privileges are usually granted only to administrators:

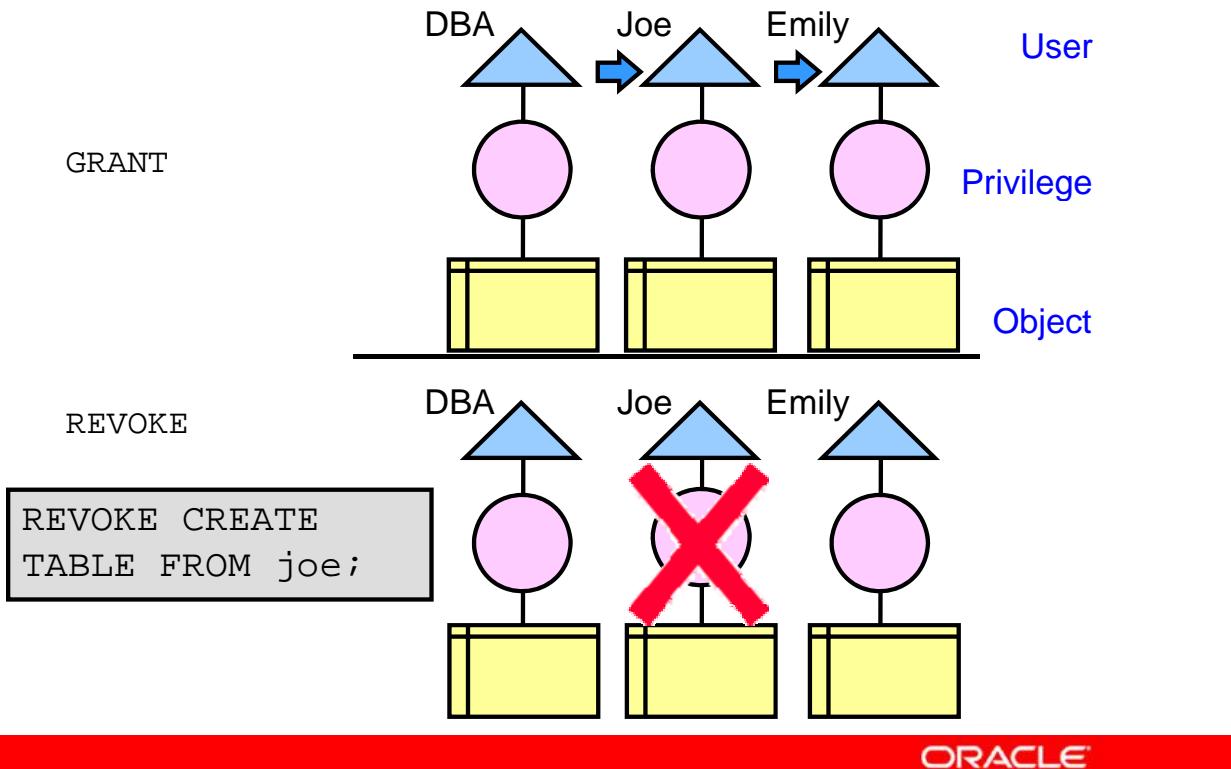
- **RESTRICTED SESSION:** This privilege allows you to log in even if the database has been opened in restricted mode.

- **SYSDBA and SYSOPER:** These privileges allow you to shut down, start up, and perform recovery and other administrative tasks in the database. SYSOPER allows a user to perform basic operational tasks, but without the ability to look at user data. It includes the following system privileges:
 - STARTUP and SHUTDOWN
 - CREATE SPFILE
 - ALTER DATABASE OPEN/MOUNT/BACKUP
 - ALTER DATABASE ARCHIVELOG
 - ALTER DATABASE RECOVER (Complete recovery only. Any form of incomplete recovery, such as UNTIL TIME | CHANGE | CANCEL | CONTROLFILE, requires connecting as SYSDBA.)
 - RESTRICTED SESSION

The SYSDBA system privilege additionally authorizes incomplete recovery and the deletion of a database. Effectively, the SYSDBA system privilege allows a user to connect as the SYS user.

- **SYSASM:** This privilege allows you to start up, shut down, and administer an ASM instance.
- **DROP ANY object:** The DROP ANY privilege allows you to delete objects that other schema users own.
- **CREATE, MANAGE, DROP, and ALTER TABLESPACE:** These privileges allow for tablespace administration, including creating, dropping, and changing tablespace attributes.
- **CREATE LIBRARY:** The Oracle database allows developers to create and call external code (for example, a C library) from PL/SQL. The library must be named by a LIBRARY object in the database. The CREATE LIBRARY privilege allows a user to create an arbitrary code library that is executable from PL/SQL.
- **CREATE ANY DIRECTORY:** As a security measure, the operating system directory where the code resides must be linked to a virtual Oracle directory object. With the CREATE ANY DIRECTORY privilege, you can potentially call insecure code objects.
The CREATE ANY DIRECTORY privilege allows a user to create a directory object (with read and write access) to any directory that the Oracle software owner can access. This means that the user can access external procedures in those directories. The user can attempt to directly read and write any database file, such as data files, redo log, and audit logs. Ensure that your organization has a security strategy that prevents misuse of powerful privileges such as this one.
- **GRANT ANY OBJECT PRIVILEGE:** This privilege allows you to grant object permissions on objects that you do not own.
- **ALTER DATABASE and ALTER SYSTEM:** These very powerful privileges allow you to modify the database and the Oracle instance (for example, renaming a data file or flushing the buffer cache).

Revoking System Privileges with ADMIN OPTION



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

System privileges that have been granted directly with a GRANT command can be revoked by using the REVOKE SQL statement. Users with ADMIN OPTION for a system privilege can revoke the privilege from any other database user. The revoker does not have to be the same user who originally granted the privilege.

There are no cascading effects when a system privilege is revoked, regardless of whether it is given the ADMIN OPTION.

The SQL syntax for revoking system privileges is:

```
REVOKE <system_privilege> FROM <grantee_clause>
```

The slide illustrates the following situation.

Scenario

1. The DBA grants the CREATE TABLE system privilege to Joe with ADMIN OPTION.
2. Joe creates a table.
3. Joe grants the CREATE TABLE system privilege to Emily.
4. Emily creates a table.
5. The DBA revokes the CREATE TABLE system privilege from Joe.

Result

Joe's table still exists, but Joe cannot create new tables. Emily's table still exists, and she still has the CREATE TABLE system privilege.

Granting Object Privileges

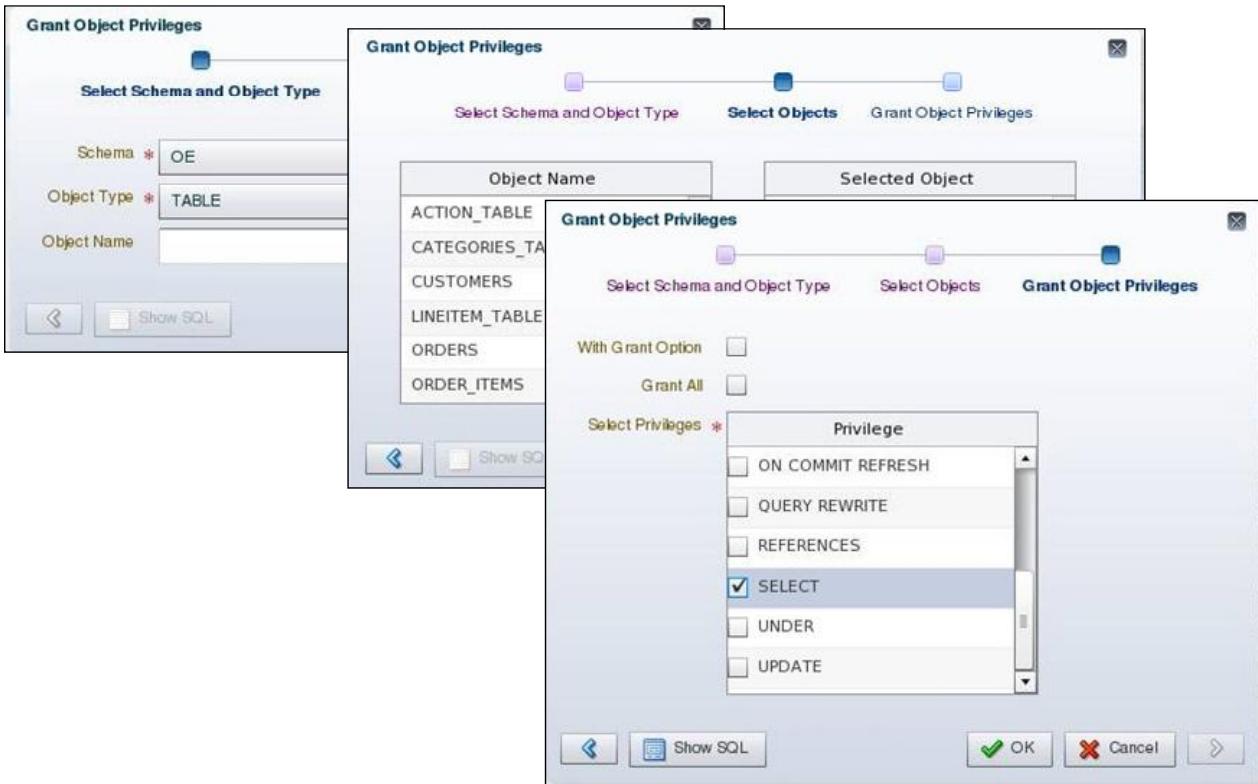
The screenshot shows the Oracle Enterprise Manager Database Express 12c interface. The title bar reads "ORACLE Enterprise Manager Database Express 12c". Below it is a navigation bar with tabs: Configuration, Storage, Security, and Performance. The main area is titled "Users". A context menu is open over the "HR" user row, listing options: Create User, Create Like, Drop User, View Details, Alter Account, Alter Tablespaces, Alter Privileges & Roles, and Grant Object Privileges. The "Grant Object Privileges" option is highlighted with a blue selection bar.

	Account Status	Expiration Date	Default Tablespace
HR	✓	Mon Apr 6, 2015 11:58:11	USERS
IX	⌚🔒	Thu Oct 2, 2014 5:44:11	USERS
LBACSYS	⌚🔒	Mon Jul 7, 2014 6:52:34	SYSTEM

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To grant object privileges, select the user on the Users page. Select “Grant Object Privileges” in the Actions menu.

Object Privileges



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

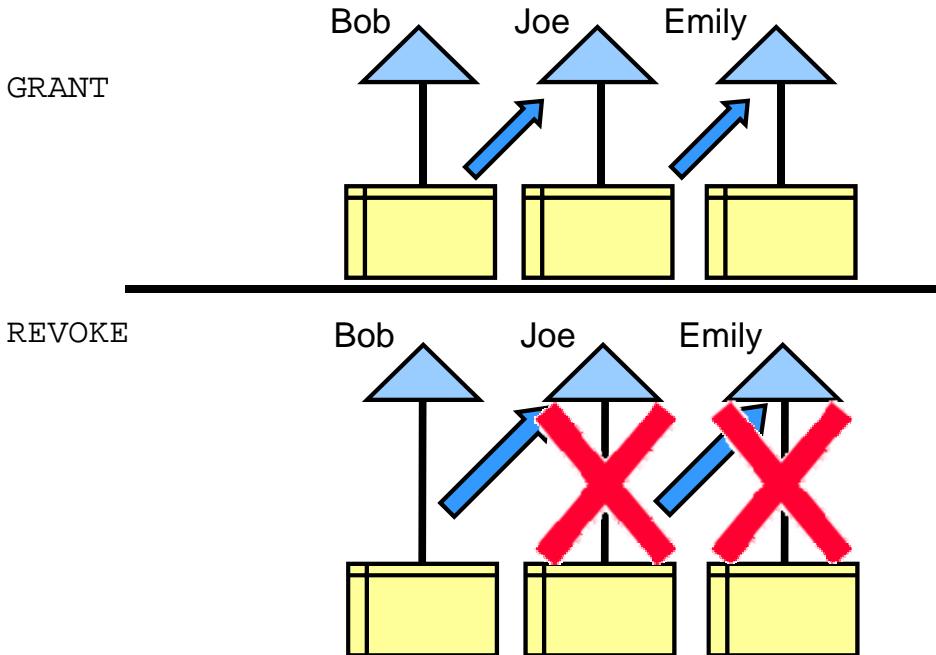
Perform the following steps to grant object privileges by using Enterprise Manager Database Express:

1. Specify the schema name for the object and select the type of object on which you want to grant privileges. Click the arrow to proceed.
2. Select the objects from the Object Name list and move them to the Selected Object list. When you have selected all objects, click the arrow to move to the next page.
3. Then select the appropriate privileges from the Privileges list. Select the "With Grant Option" check box if this user is allowed to grant other users the same access.

The SQL syntax for granting object privileges is:

```
GRANT <object_privilege> ON <object> TO <grantee_clause>
[WITH GRANT OPTION]
```

Revoking Object Privileges with GRANT OPTION



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Cascading effects can be observed when revoking a system privilege that is related to a data manipulation language (DML) operation. For example, if the SELECT ANY TABLE privilege is granted to a user, and if that user has created procedures that use the table, all procedures that are contained in the user's schema must be recompiled before they can be used again.

Revoking object privileges also cascades when given with GRANT OPTION. As a user, you can revoke only those privileges that you have granted. For example, Bob cannot revoke the object privilege that Joe granted to Emily. Only the grantee or a user with the privilege called GRANT ANY OBJECT PRIVILEGE can revoke object privileges.

Scenario

1. Joe is granted the SELECT object privilege on EMPLOYEES with GRANT OPTION.
2. Joe grants the SELECT privilege on EMPLOYEES to Emily.
3. The SELECT privilege is revoked from Joe. This revoke is cascaded to Emily as well.

Using Roles to Manage Privileges

- Roles:
 - Used to group together privileges and roles
 - Facilitate granting of multiple privileges or roles to users
- Benefits of roles:
 - Easier privilege management
 - Dynamic privilege management
 - Selective availability of privileges



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

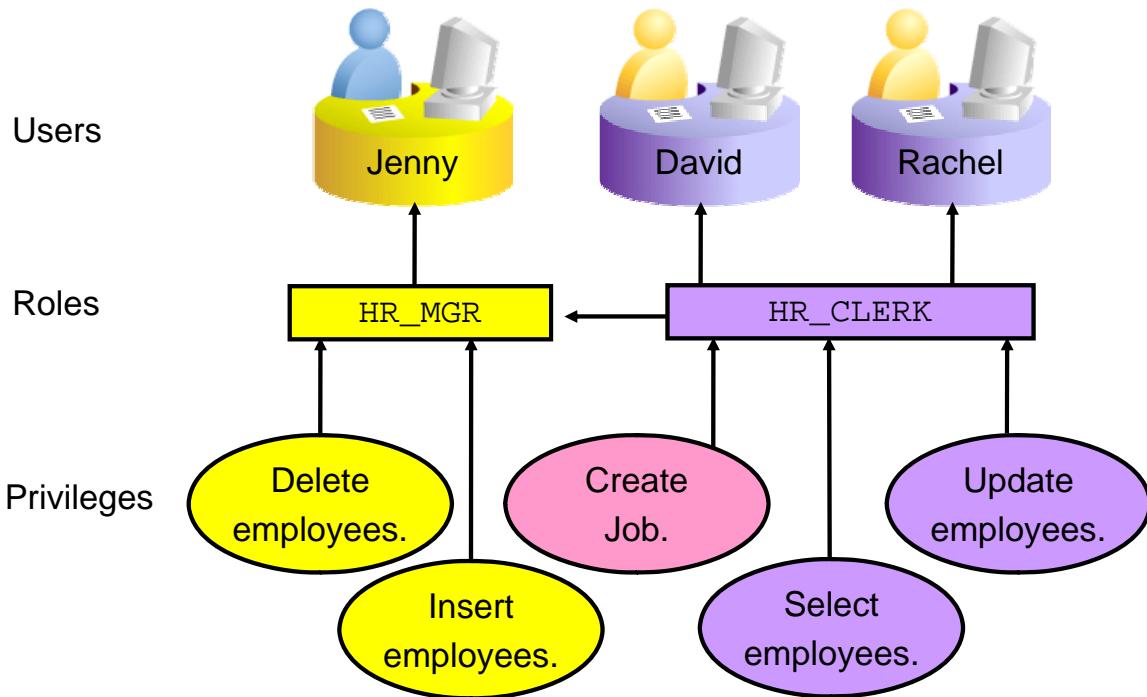
A role is a named group of related privileges that are granted to users or to other roles.

You can use roles to administer database privileges. You can add privileges to a role and grant the role to a user. The user can then enable the role and exercise the privileges granted by the role. A role contains all privileges that are granted to that role and all privileges of other roles that are granted to it.

Roles provide the following benefits with respect to managing privileges:

- **Easier privilege management:** Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role and then grant that role to each user.
- **Dynamic privilege management:** If the privileges associated with a role are modified, all users who are granted the role acquire the modified privileges automatically and immediately.
- **Selective availability of privileges:** Roles can be enabled and disabled to turn privileges on and off temporarily. This allows the privileges of the user to be controlled in a given situation.

Assigning Privileges to Roles and Assigning Roles to Users



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In most systems, it is time-consuming and error-prone to grant necessary privileges to each user individually. Oracle software provides for easy and controlled privilege management through roles. Roles are named groups of related privileges that are granted to users or to other roles. Roles are designed to ease the administration of privileges in the database and, therefore, improve security.

Role characteristics

- Privileges are granted to and revoked from roles as though the role were a user.
- Roles are granted to and revoked from users or other roles as though they were system privileges.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Roles are not owned by anyone, and they are not in any schema.

In the slide example, the SELECT and UPDATE privileges on the employees table and the CREATE JOB system privilege are granted to the HR_CLERK role. DELETE and INSERT privileges on the employees table and the HR_CLERK role are granted to the HR_MGR role. The manager is granted the HR_MGR role and can now select, delete, insert, and update the employees table.

Predefined Roles

Role	Privileges Included
CONNECT	CREATE SESSION
DBA	Most system privileges; several other roles. Do not grant to non-administrators.
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
SCHEDULER_ ADMIN	CREATE ANY JOB, CREATE EXTERNAL JOB, CREATE JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER
SELECT_CATALOG_ROLE	SELECT privileges on data dictionary objects



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are several roles that are defined automatically for Oracle databases when you run database creation scripts. CONNECT is granted automatically to any user that is created with Enterprise Manager.

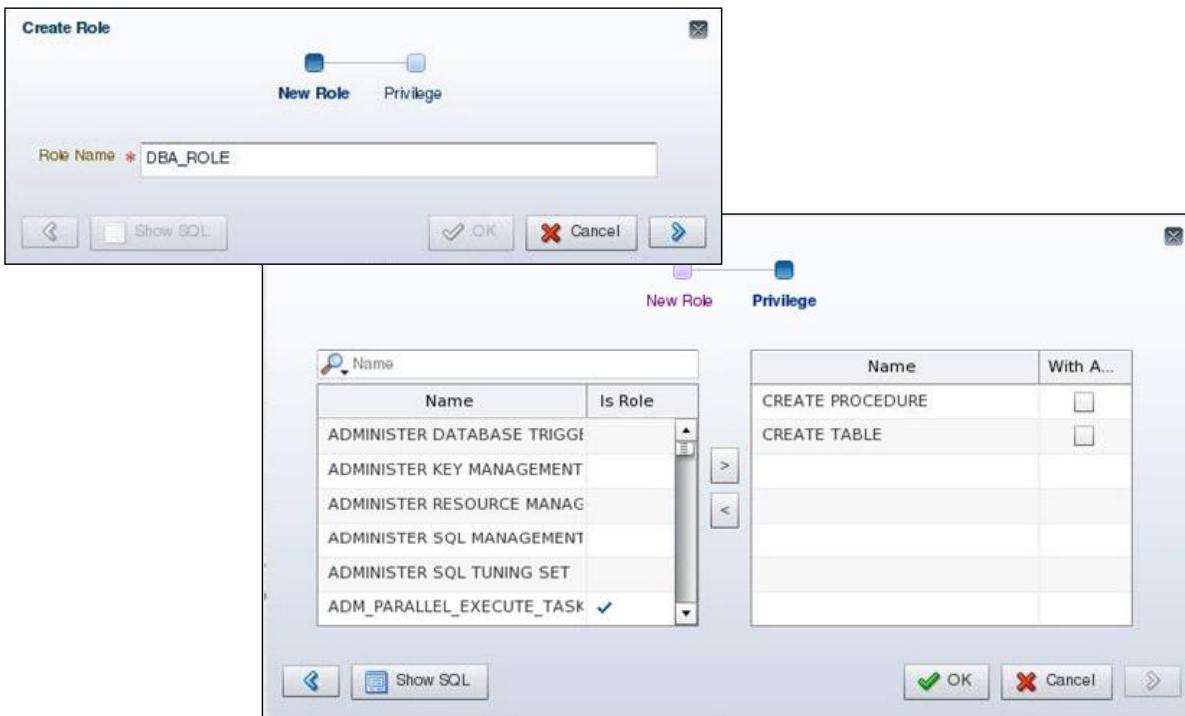
The SELECT ANY DICTIONARY system privilege does not permit access to sensitive data dictionary tables, which are owned by the SYS schema.

Refer to the *Oracle Database Security Guide* for a complete list of predefined roles.

Other roles that authorize you to administer special functions are created when that functionality is installed. For example, XDBADMIN contains the privileges required to administer the Extensible Markup Language (XML) database if that feature is installed. AQ_ADMINISTRATOR_ROLE provides privileges to administer advanced queuing. HS_ADMIN_ROLE includes the privileges needed to administer heterogeneous services.

You must not alter the privileges granted to these functional roles without the assistance of Oracle Support because you may inadvertently disable the needed functionality.

Creating a Role



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a role by using Enterprise Manager Database Express, perform the following steps:

1. Navigate to the Roles page and click Create Role.
2. Enter a name for the role.
3. Optionally add the system privileges and other roles as required. The role can be edited at a later time to modify these settings if necessary. You can also add object privileges to a role after it is created.
4. Click OK to create the role.

Secure Roles

- Roles can be nondefault and enabled when required.

```
SET ROLE vacationdba;
```

- Roles can be protected through authentication.
- Roles can also be secured programmatically.

```
CREATE ROLE secure_application_role  
IDENTIFIED USING <security_procedure_name>;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

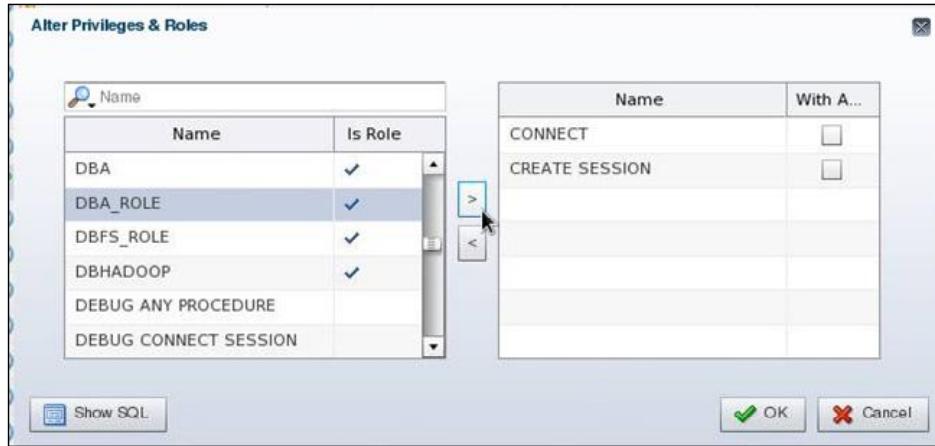
Roles are usually enabled by default, which means that if a role is granted to a user, then that user can exercise the privileges given to the role. Default roles are assigned to the user at connect time.

It is possible to:

- Make a role nondefault. The user must now explicitly enable the role before the role's privileges can be exercised.
- Have a role require additional authentication by using the IDENTIFIED clause to indicate that a user must be authorized by a specified method before the role is enabled with the SET ROLE statement. The default authentication for a role is None.
- Create secure application roles that can be enabled only by executing a PL/SQL procedure successfully. The PL/SQL procedure can check things such as the user's network address, the program that the user is running, the time of day, and other elements needed to properly secure a group of permissions.
- Administer roles easily using the Oracle Database Vault option. Secure application roles are simplified, and traditional roles can be further restricted.

Note: Role authentication can be defined by using Enterprise Manager Cloud Control, but not in Enterprise Manager Database Express.

Assigning Roles to Users



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

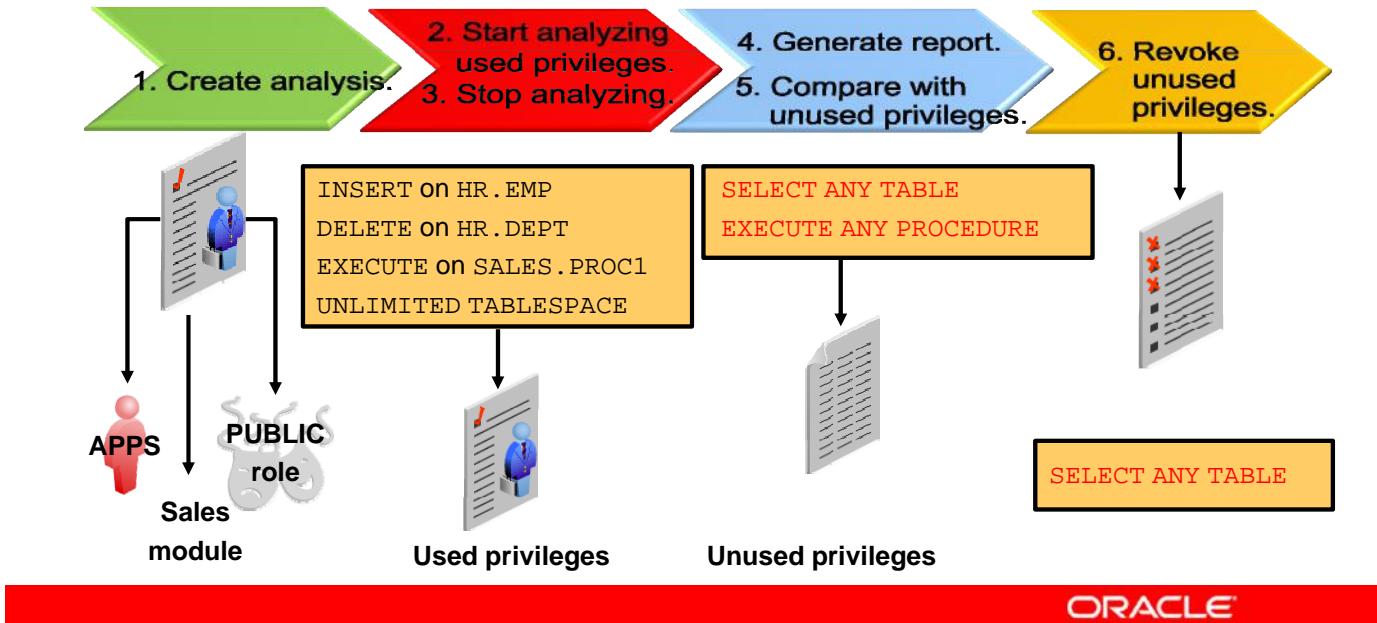
By default, Enterprise Manager Database Express automatically grants the CONNECT role to new users. This allows users to connect to the database and create database objects in their own schemas.

To assign a role to a user by using Enterprise Manager Database Express, perform the following steps:

1. Navigate to the Users page.
2. Select the user and select “Alter Privileges & Roles” in the Actions menu.
3. Select the desired role in the list on the left and move it by using the right arrow.
4. When you have assigned all appropriate roles, click OK.

Privilege Analysis

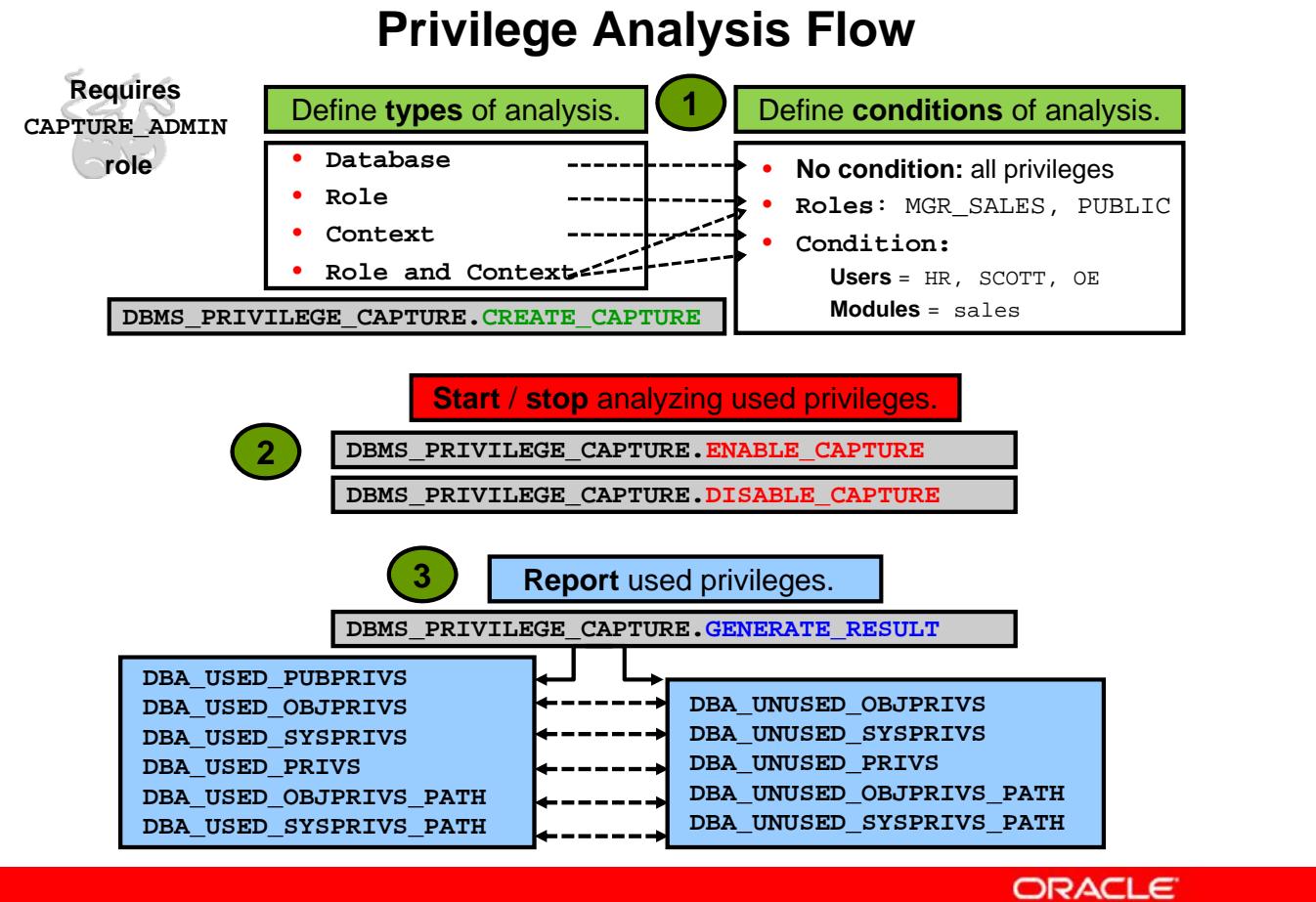
- Analyze used privileges to revoke unnecessary privileges.
- Use DBMS_PRIVILEGE_CAPTURE package.



A major concern in many databases 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.

Oracle Database 12c provides a package named `DBMS_PRIVILEGE_CAPTURE` 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 privilege lists.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

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

Profiles and Users

Users are assigned only one profile at a time.

Profiles:

- Control resource consumption
- Manage account status and password expiration



Note: RESOURCE_LIMIT must be set to TRUE before profiles can impose resource limitations.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Profiles impose a named set of resource limits on database usage and instance resources. Profiles also manage the account status and place limitations on users' passwords (length, expiration time, and so on). Every user is assigned a profile and may belong to only one profile at any given time. If users have already logged in when you change their profile, the change does not take effect until their next login.

The DEFAULT profile serves as the basis for all other profiles. As illustrated in the slide, limitations for a profile can be implicitly specified (as in CPU/Session), can be unlimited (as in CPU/Call), or can reference whatever setting is in the DEFAULT profile (as in Connect Time).

Profiles cannot impose resource limitations on users unless the RESOURCE_LIMIT initialization parameter is set to TRUE. With RESOURCE_LIMIT at its default value of FALSE, profile resource limitations are ignored. Profile password settings are always enforced.

Profiles enable an administrator to control the following system resources:

- **CPU:** CPU resources may be limited on a per-session or per-call basis. A CPU/Session limitation of 1,000 means that if any individual session that uses this profile consumes more than 10 seconds of CPU time (CPU time limitations are in hundredths of a second), that session receives an error and is logged off:

ORA-02392: exceeded session limit on CPU usage, you are being logged off

A per-call limitation does the same thing, but instead of limiting the user's overall session, it prevents any single command from consuming too much CPU. If CPU/Call is limited and the user exceeds the limitation, the command aborts. The user receives an error message such as the following:

ORA-02393: exceeded call limit on CPU usage

- **Network/Memory:** Each database session consumes system memory resources and (if the session is from a user who is not local to the server) network resources. You can specify the following:
 - **Connect Time:** Indicates for how many minutes a user can be connected before being automatically logged off
 - **Idle Time:** Indicates for how many minutes a user's session can remain idle before being automatically logged off. Idle time is calculated for the server process only. It does not take into account application activity. The `IDLE_TIME` limit is not affected by long-running queries and other operations.
 - **Concurrent Sessions:** Indicates how many concurrent sessions can be created by using a database user account
 - **Private SGA:** Limits the amount of space consumed in the System Global Area (SGA) for sorting, merging bitmaps, and so on. This restriction takes effect only if the session uses a shared server. (Shared servers are covered in the lesson titled "Configuring the Oracle Network Environment.")
- **Disk I/O:** This limits the amount of data a user can read at the per-session level or per-call level. Reads/Session and Reads/Call place a limitation on the total number of reads from both memory and the disk. This can be done to ensure that no I/O-intensive statements overuse memory and disks.

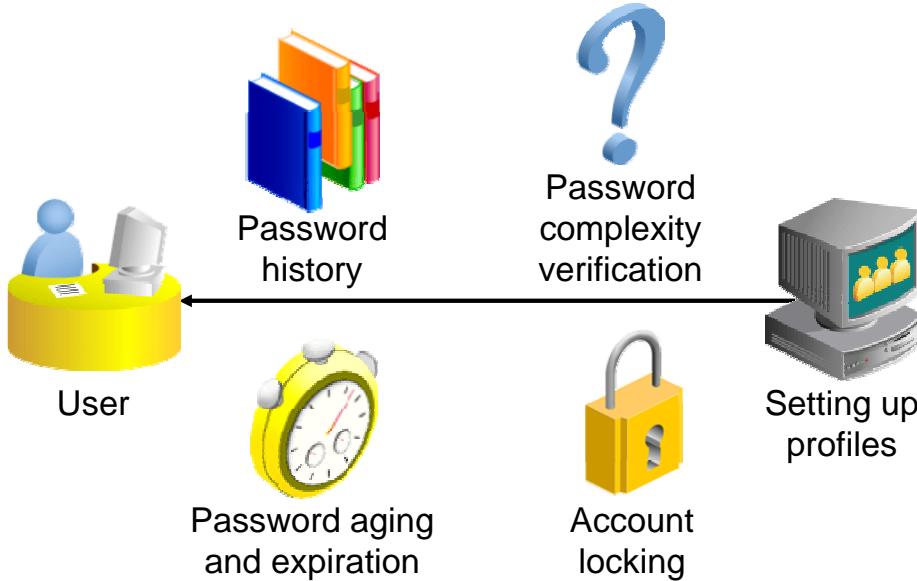
Profiles also allow a composite limit. Composite limits are based on a weighted combination of CPU/Session, Reads/Session, Connect Time, and Private SGA. Composite limits are discussed in more detail in the *Oracle Database Security Guide*.

To create a profile by using Enterprise Manager Database Express:

1. Select Profiles in the Security menu.
2. Select Create Profile.
3. Enter the profile name and then click the arrow to continue.
4. On the General page, enter the values for the resources you want to define. Click the arrow to continue.
5. On the Password page, enter the appropriate values.
6. Click OK to create the profile.

Note: Resource Manager is an alternative to many of the profile settings. For more details about Resource Manager, see the *Oracle Database Administrator's Guide*.

Implementing Password Security Features



Note: Do not use profiles that cause the SYS, SYSMAN, and DBSNMP passwords to expire and the accounts to be locked.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle password management is implemented with user profiles. Profiles can provide many standard security features.

Account locking: Enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts

- **FAILED_LOGIN_ATTEMPTS:** Specifies the number of failed login attempts before the lockout of the account
- **PASSWORD_LOCK_TIME:** Specifies the number of days for which the account is locked after the specified number of failed login attempts

Password aging and expiration: Enables user passwords to have a lifetime, after which the passwords expire and must be changed

- **PASSWORD_LIFE_TIME:** Determines the lifetime of the password in days, after which the password expires
- **PASSWORD_GRACE_TIME:** Specifies a grace period in days for changing the password after the first successful login after the password has expired

Note: Expiring passwords and locking the SYS, SYSMAN, and DBSNMP accounts prevent Enterprise Manager from functioning properly. The applications must catch the “password expired” warning message and handle the password change; otherwise, the grace period expires and the user is locked out without knowing the reason.

Password history: Checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes. These checks can be implemented by using one of the following:

- **PASSWORD_REUSE_TIME:** Specifies that a user cannot reuse a password for a given number of days
- **PASSWORD_REUSE_MAX:** Specifies the number of password changes that are required before the current password can be reused

Recall that the values of the profile parameters are either set or inherited from the `DEFAULT` profile.

If both password history parameters have a value of `UNLIMITED`, Oracle Database ignores both. The user can reuse any password at any time, which is not a good security practice.

If both parameters are set, password reuse is allowed—but only after meeting both conditions. The user must have changed the password the specified number of times, and the specified number of days must have passed since the old password was last used.

For example, the profile of user `ALFRED` has `PASSWORD_REUSE_MAX` set to 10 and `PASSWORD_REUSE_TIME` set to 30. User `ALFRED` cannot reuse a password until he has reset the password 10 times and until 30 days have passed since the password was last used.

If one parameter is set to a number and the other parameter is specified as `UNLIMITED`, then the user can never reuse a password.

Password complexity verification: Makes a complexity check on the password to verify that it meets certain rules. The check must ensure that the password is complex enough to provide protection against intruders who may try to break into the system by guessing the password.

The `PASSWORD_VERIFY_FUNCTION` parameter names a PL/SQL function that performs a password complexity check before a password is assigned. Password verification functions must be owned by the `SYS` user and must return a Boolean value (`TRUE` or `FALSE`). A model password verification function is provided in the `ut1pwdmg.sql` script found in the following directories:

- UNIX and Linux platforms: `$ORACLE_HOME/rdbms/admin`
- Windows platforms: `%ORACLE_HOME%\rdbms\admin`

Creating a Password Profile



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can choose common values for each of the settings from a list of values (click the flashlight icon to browse), or you can enter a custom value.

All time periods are expressed in days but can also be expressed as fractions. There are 1,440 minutes in a day; 5/1,440 is therefore five minutes.

Enterprise Manager can also be used to edit existing password profiles.

If the `utlpwdmg.sql` script has been executed, the `VERIFY_FUNCTION_11g`, `ORA12C_VERIFY_FUNCTION`, and `ORA12C_STRONG_VERIFY_FUNCTION` functions are available. Additional information about these functions is provided later in the lesson. If you have created your own complexity function, the name of that function may be entered. The function name does not appear in the Select list. If the function produces runtime errors, the user is unable to change the password.

Dropping a Password Profile

In Enterprise Manager, you cannot drop a profile that is used by users. However, if you drop a profile with the `CASCADE` option (for example, in SQL*Plus), all users who have that profile are automatically assigned the `DEFAULT` profile.

Supplied Password Verification Functions

- The following functions are created by the \$ORACLE_HOME/rdbms/admin/utlpwdmg.sql script:
 - VERIFY_FUNCTION_11g
 - ORA12C_VERIFY_FUNCTION
 - ORA12C_STRONG_VERIFY_FUNCTION
- The functions require the following of passwords:
 - Have a minimum number of characters
 - Not be the username, username with a number, or username reversed
 - Not be the database name or the database name with a number
 - Have at least one alphabetic and one numeric character
 - Differ from the previous password by at least three letters



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can create the following password verification functions by executing the utlpwdmg.sql script:

- ORA12C_VERIFY_FUNCTION: This function makes the minimum complexity checks such as checking for a minimum password length and that the password is not the same as the username.
- ORA12C_STRONG_VERIFY_FUNCTION: This function provides a stronger password complexity function that takes into consideration recommendations from the US Department of Defense Database Security Technical Implementation Guide.
- VERIFY_FUNCTION_11g: This function makes minimum complexity checks such as checking for a minimum password length and that the password is not the same as the username. This function was provided with Oracle Database 11g.

Note: The functions must be owned by the SYS user. Password complexity checking is not enforced for the SYS user.

Refer to the *Oracle Database Security Guide* for a detailed description of each of the functions.

Modifications to the Default Profile

The `utlpwdmg.sql` script also modifies the `DEFAULT` profile as follows:

```
ALTER PROFILE DEFAULT LIMIT  
PASSWORD_LIFE_TIME 180  
PASSWORD_GRACE_TIME 7  
PASSWORD_REUSE_TIME UNLIMITED  
PASSWORD_REUSE_MAX UNLIMITED  
FAILED_LOGIN_ATTEMPTS 10  
PASSWORD_LOCK_TIME 1  
PASSWORD_VERIFY_FUNCTION  
    ora12c_verify_function;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In addition to creating the functions as shown on the previous page, the `utlpwdmg` script also changes the `DEFAULT` profile with the following `ALTER PROFILE` command:

```
ALTER PROFILE default LIMIT  
PASSWORD_LIFE_TIME 180  
PASSWORD_GRACE_TIME 7  
PASSWORD_REUSE_TIME UNLIMITED  
PASSWORD_REUSE_MAX UNLIMITED  
FAILED_LOGIN_ATTEMPTS 10  
PASSWORD_LOCK_TIME 1  
PASSWORD_VERIFY_FUNCTION ora12c_verify_function;
```

Remember that when users are created, they are assigned the `DEFAULT` profile unless another profile is specified.

Assigning Quotas to Users

- Users who do not have the **UNLIMITED TABLESPACE** system privilege must be given a quota before they can create objects in a tablespace.
- Quotas can be:
 - A specific value in megabytes or kilobytes
 - Unlimited



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A **quota** is a space allowance in a given tablespace. By default, a user has no quota on any of the tablespaces. You have three options for providing a quota for a user on a tablespace.

- **Unlimited:** Allows the user to use as much space as is available in the tablespace
- **Value:** Number of kilobytes or megabytes that the user can use. This does not guarantee that the space is set aside for the user. This value can be larger or smaller than the current space that is available in the tablespace.
- **UNLIMITED TABLESPACE system privilege:** Overrides all individual tablespace quotas and gives the user unlimited quota on all tablespaces, including **SYSTEM** and **SYSAUX**. This privilege must be granted with caution.

You must not provide a quota to users on the **SYSTEM** or **SYSAUX** tablespaces. Typically, only the **SYS** and **SYSTEM** users are able to create objects in the **SYSTEM** or **SYSAUX** tablespaces.

You do not need a quota on an assigned temporary tablespace or any undo tablespaces. You do not need to have a quota to insert, update, and delete data in an Oracle database. The only users that need quota are the accounts that own the database objects. It is typical when installing application code that the installer creates database accounts to own the objects. Only these accounts need quotas. Other database users can be granted permission to use these objects without a quota.

- The Oracle server checks the quota when a user creates or extends a segment.
- For activities that are assigned to a user schema, only those activities that use space in a tablespace count against the quota. Activities that do not use space in the assigned tablespace do not affect the quota (such as creating views or using temporary tablespaces).
- The quota is replenished when objects owned by the user are dropped with the PURGE clause or when the objects owned by the user in the recycle bin are purged.

Applying the Principle of Least Privilege

- Protect the data dictionary:

```
O7_DICTIONARY_ACCESSIBILITY=FALSE
```

- Revoke unnecessary privileges from PUBLIC.
- Use access control lists (ACL) to control network access.
- Restrict the directories accessible by users.
- Limit users with administrative privileges.
- Restrict remote database authentication:

```
REMOTE_OS_AUTHENT=FALSE
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The principle of least privilege means that a user must be given only those privileges that are required to efficiently complete a task. This reduces the chances of users modifying or viewing data (either accidentally or maliciously) that they do not have the privilege to modify or view.

Protect the data dictionary: The O7_DICTIONARY_ACCESSIBILITY parameter is set by default to FALSE. You must not allow this to be changed without a very good reason because it prevents users with the ANY TABLE system privileges from accessing the data dictionary base tables. It also ensures that the SYS user can log in only as SYSDBA.

Revoke unnecessary privileges from PUBLIC: Several packages are extremely useful to applications that need them, but require proper configuration to be used securely. PUBLIC is granted execute privilege on the following packages: UTL_SMTP, UTL_TCP, UTL_HTTP, and UTL_FILE.

Control network access: In Oracle Database, network access is controlled by an access control list (ACL) that may be configured to allow certain users access to specific network services. Network access is denied by default. An ACL must be created to allow network access. File access through UTL_FILE is controlled at two levels:

- At the OS level with permissions on files and directories
- In the database by DIRECTORY objects that allow access to specific file system directories. The DIRECTORY object may be granted to a user for read or for read and write. Execute privileges on other PL/SQL packages should be carefully controlled.

The more powerful packages that may potentially be misused include:

- **UTL_SMTP:** Permits arbitrary email messages to be sent by using the database as a Simple Mail Transfer Protocol (SMTP) mail server. Use the ACL to control which machines may be accessed by which users.
- **UTL_TCP:** Permits outgoing network connections to be established by the database server to any receiving or waiting network service. Thus, arbitrary data can be sent between the database server and any waiting network service. Use the ACL to control access.
- **UTL_HTTP:** Allows the database server to request and retrieve data via HTTP. Granting this package to a user may permit data to be sent via HTML forms to a malicious website. Limit access by using the ACL.
- **UTL_FILE:** If configured improperly, allows text-level access to any file on the host operating system. When properly configured, this package limits user access to specific directory locations.

Restrict access to OS directories: The DIRECTORY object inside the database enables DBAs to map directories to OS paths and to grant privileges on those directories to individual users.

Limit users with administrative privileges: Do not provide database users more privileges than necessary. Non-administrators must not be granted the DBA role. To implement least privilege, restrict the following types of privileges:

- Grants of system and object privileges
- SYS-privileged connections to the database, such as SYSDBA and SYSOPER
- Other DBA-type privileges, such as DROP ANY TABLE

Restrict remote database authentication: The REMOTE_OS_AUTHENT parameter is set to FALSE by default. It must not be changed unless all clients can be trusted to authenticate users appropriately.

In the remote authentication process:

- The database user is authenticated externally
- The remote system authenticates the user
- The user logs in to the database without further authentication

Note: Always test your applications thoroughly if you have revoked privileges.

Quiz

All passwords created in Oracle Database are not case-sensitive by default.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

A database role:

- a. Can be enabled or disabled
- b. Can consist of system and object privileges
- c. Is owned by its creator
- d. Cannot be protected by a password



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Quiz

With RESOURCE_LIMIT set at its default value of FALSE, profile password limitations are ignored.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

Applying the principle of least privilege is not enough to harden the Oracle database.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Create and manage database user accounts:
 - Authenticate users
 - Assign default storage areas (tablespaces)
- Grant and revoke privileges
- Create and manage roles
- Create and manage profiles:
 - Implement standard password security features
 - Control resource usage by users



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Creating a profile to limit resource consumption
- Creating two roles:
 - HRCLERK
 - HRMANAGER
- Creating four new users:
 - One manager and two clerks
 - One schema user for the next practice session



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Oracle University and Error : You are not a Valid Partner use only

Managing Database Storage Structures



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

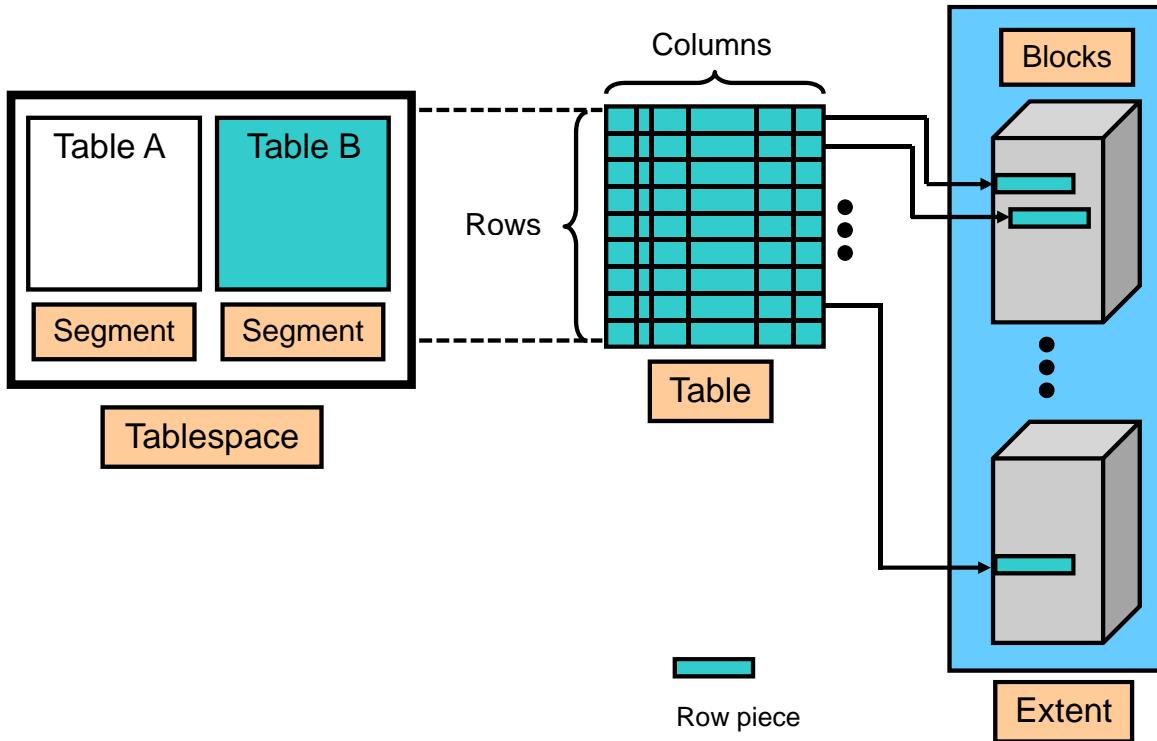
After completing this lesson, you should be able to:

- Describe the storage of table row data in blocks
- Create and manage tablespaces
- Obtain tablespace information



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

How Table Data Is Stored



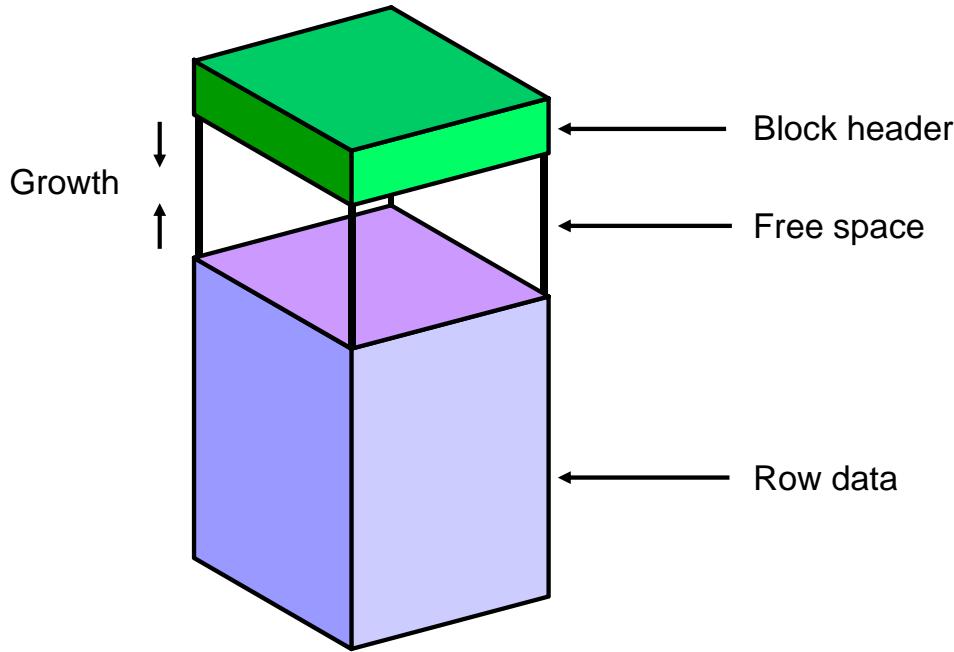
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When a table is created, a segment is created to hold its data. A tablespace contains a collection of segments.

Logically, a table contains rows of column values. A row is ultimately stored in a database block in the form of a row piece. It is called a row piece because, under some circumstances, the entire row may not be stored in one place. This happens when an inserted row is too large to fit into a single block (chained row) or when an update causes an existing row to outgrow the available free space of the current block (migrated row). Row pieces are also used when a table has more than 255 columns. In this case the pieces may be in the same block (intra-block chaining) or across multiple blocks.

Database Block: Contents



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **Block header:** The block header contains the segment type (such as table or index), data block address, table directory, row directory, and transaction slots of approximately 23 bytes each, which are used when modifications are made to rows in the block. The block header grows downward from the top.
- **Row data:** This is the actual data for the rows in the block. Row data space grows upward from the bottom.
- **Free space:** Free space is in the middle of the block, enabling the header and the row data space to grow when necessary. Row data takes up free space as new rows are inserted or as columns of existing rows are updated with larger values.

Examples of events that cause header growth:

- Row directories that need more row entries
- More transaction slots required than initially configured

Initially, the free space in a block is contiguous. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the Oracle server when necessary.

Exploring the Storage Structure

The screenshot shows two views of the Oracle Enterprise Manager Database Express 12c interface. The top view is the 'Database Home' page, which includes a 'Status' section with database details like Up Time, Type, Version, Database Name, Instance Name, and Platform Name. The bottom view is the 'Tablespaces' page, which displays a grid of tablespace information. The 'Tablespaces' page includes columns for Name, Size, Free Space, Used (%), Autoextend, Maxsize, Status, Type, Group, Auto Shrink, and Directory. The table lists six tablespaces: EXAMPLE, SYSAUX, SYSTEM, TEMP, UNDOTBS1, and USERS.

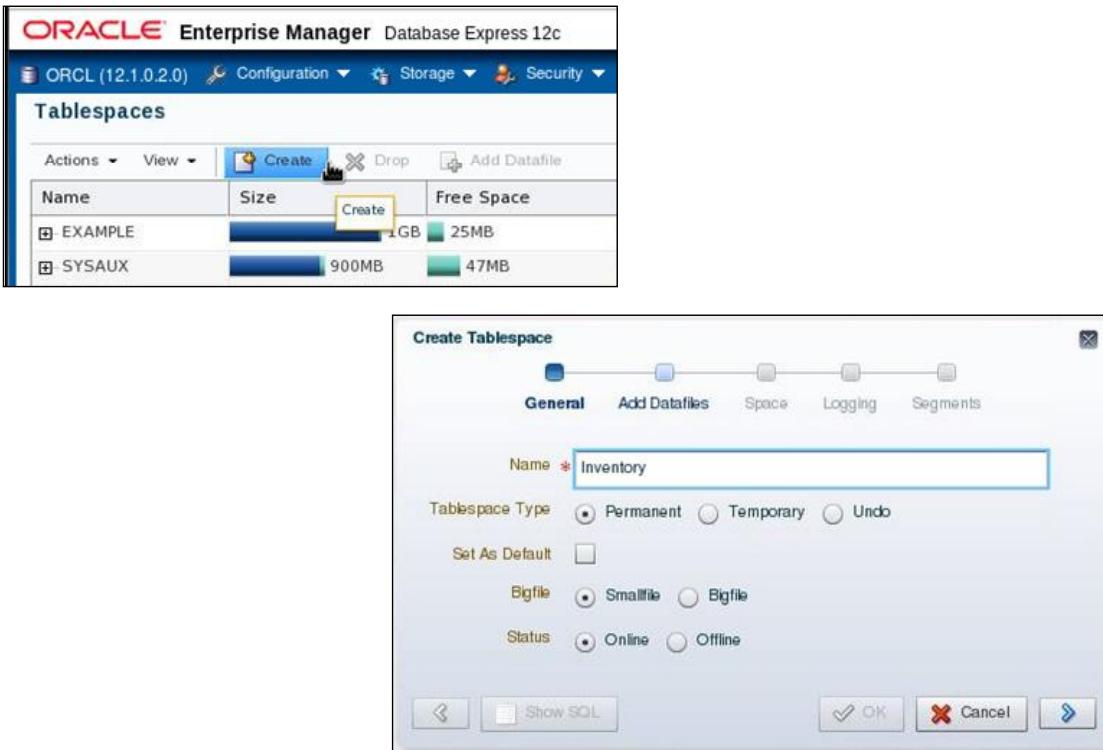
Name	Size	Free Space	Used (%)	Auto...	Max...	Status	Type	Grou...	Auto...	Directory
EXAMPLE	1GB	25MB	98	✓	Unlimi	Green	File			/u01/app/oracle/ora...
SYSAUX	900MB	47MB	94.8	✓	Unlimi	Green	File			/u01/app/oracle/ora...
SYSTEM	810MB	2MB	99.7	✓	Unlimi	Green	File			/u01/app/oracle/ora...
TEMP	197MB	195MB	1	✓	Unlimi	Green	Temporary			/u01/app/oracle/ora...
UNDOTBS1	150MB	133MB	11.5	✓	Unlimi	Green	File			/u01/app/oracle/ora...
USERS	5MB	3MB	33.8	✓	Unlimi	Green	File			/u01/app/oracle/ora...

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

You can easily view the storage structures of your database through Enterprise Manager Database Express. Detailed information about each structure can be obtained by selecting the structure in the Storage menu.

Creating a New Tablespace

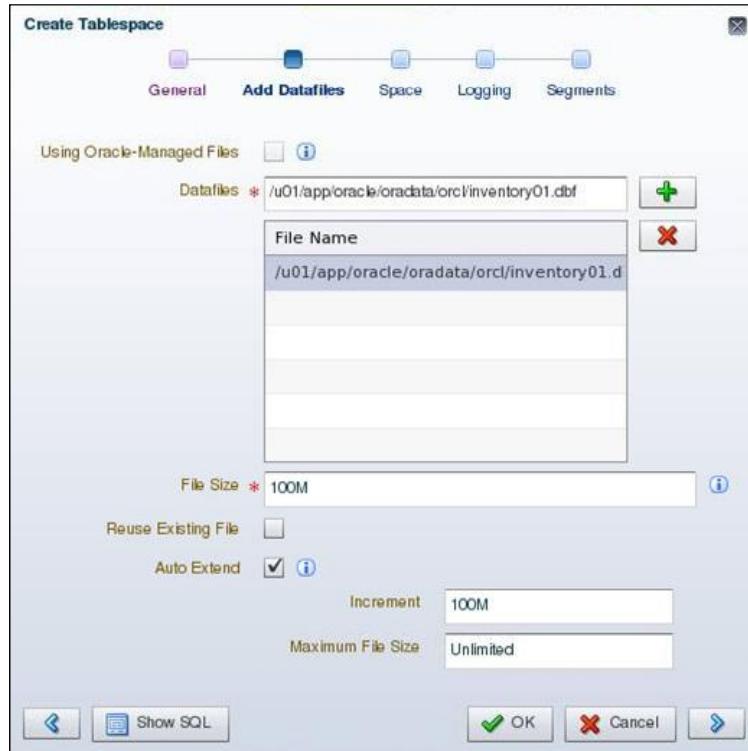


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. Expand the Storage menu, and then select Tablespaces.
2. Click Create.
3. Enter a name for the tablespace.
4. In the Tablespace Type field, select Permanent.
Permanent tablespaces store permanent database objects that are created by the system or users.
5. In the Bigfile field, select Smallfile. Bigfile tablespaces are used with extremely large databases, in which ASM or other logical volume managers support the striping or redundant array of independent disks (RAID) and dynamically extensible logical volumes.
6. In the Status field, select Online.
Online means that users can read and write to the tablespace after it is created. This is the default.
7. Click the blue arrow to add data files to the tablespace.

Creating a New Tablespace



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A tablespace must have at least one file.

8. On the Add Datafiles page, enter a file directory and file name for the data file.
9. Enter the desired file size.
10. Select “Auto Extend” to automatically extend the data file when it is full and specify an increment amount in the Increment field. This causes the data file to extend automatically each time it runs out of space. It is limited, of course, by the physical media on which it resides. Leave Maximum File Size as Unlimited or enter a maximum size.
11. Click the blue arrow to continue.

Creating a New Tablespace



ORACLE

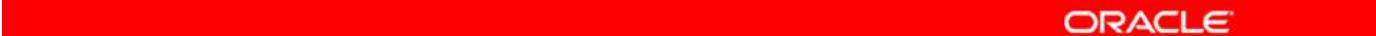
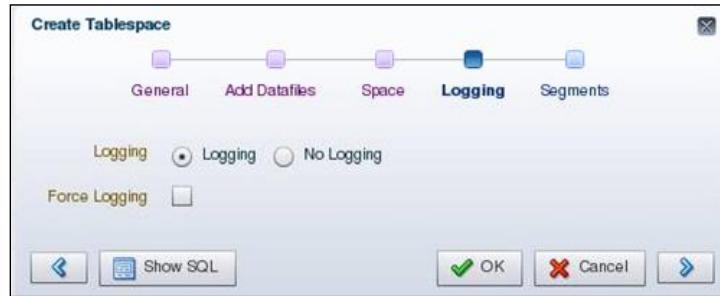
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Block Size: This field shows the block size that is used for the tablespace being created.

Extent Allocation: The extents can be allocated in one of these two ways:

- **Automatic:** Also called autoallocate, it specifies that the sizes of the extents in the tablespace are system managed. You cannot specify Automatic for a temporary tablespace.
- **Uniform:** It specifies that the tablespace is managed with uniform extents of a size that you specify. The default size is 1 MB. All extents of temporary tablespaces are uniform. You cannot specify Uniform for an undo tablespace.

Creating a New Tablespace

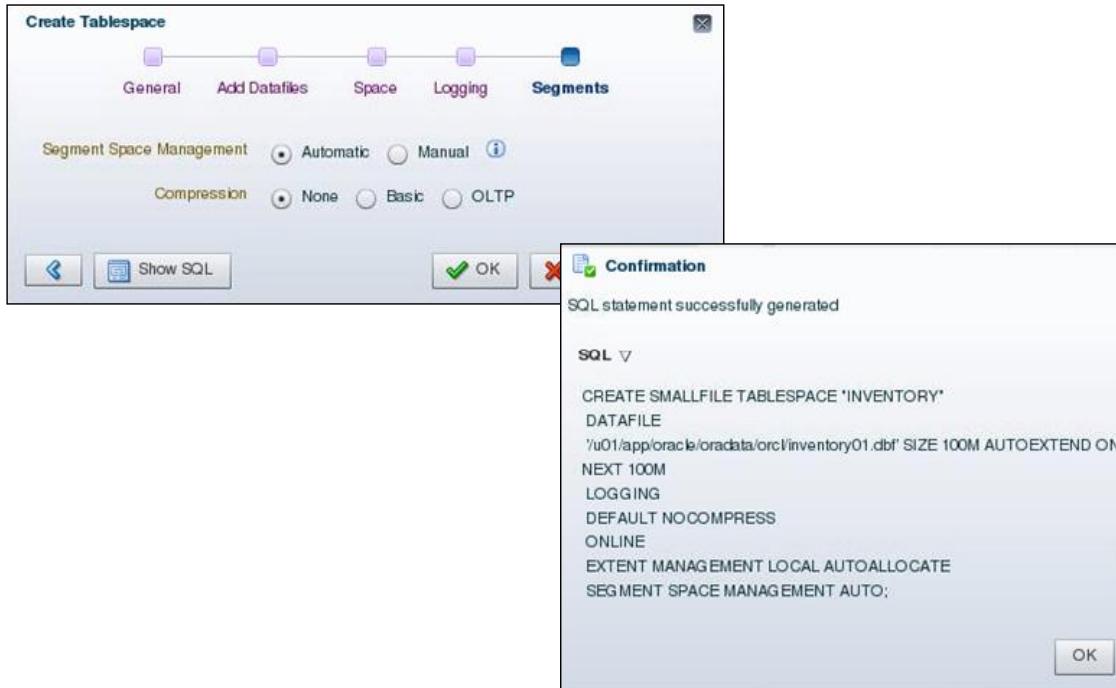
The Oracle logo, consisting of the word 'ORACLE' in white capital letters on a red background.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Logging: The `LOGGING` clause sets the default logging value for any segment created in the tablespace. Changes made to objects in the tablespace are written to the redo log. If logging is not enabled, any direct loads using SQL*Loader and direct load `INSERT` operations are not written to the redo log, and the objects are thus unrecoverable in the event of data loss. When an object is created without logging enabled, you must back up those objects if you want them to be recoverable. Choosing not to enable logging can have a significant impact on the ability to recover objects in the future. Use with caution. For more details about the logging clause, see the *Oracle Database SQL Reference*.

Note: If `FORCE LOGGING` mode is in effect for the database, it takes precedence over the tablespace logging setting. The database can be put into `FORCE LOGGING` mode at the time of database creation or after database creation using the `ALTER DATABASE FORCE LOGGING` command.

Creating a New Tablespace



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Segment Space Management: Segment space management can be specified as:

- **Automatic:** The Oracle Database server uses bitmaps to manage the free space in segments. The bitmap describes the status of each data block in a segment with respect to the amount of space in the block that is available for inserting rows. As more or less space becomes available in a data block, the new state is reflected in the bitmap. With bitmaps, the Oracle database manages free space more automatically. As a result, this form of space management is called Automatic Segment Space Management (ASSM).
- **Manual:** This specifies that you want to use free lists for managing free space in segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space in segments is called manual segment space management because of the need to specify and tune the PCTUSED, FREELISTS, and FREELIST GROUPS storage parameters for schema objects created in the tablespace. This is supported for backward compatibility; it is recommended that you use ASSM.

Compression Options: Data segment compression is disabled by default. Enabling data segment compression can save disk space usage, reduce memory use in the buffer cache, and speed up query execution during reads. There is, however, a cost in CPU overhead for data loading and DML. It is especially useful in online analytical processing (OLAP) systems, where there are lengthy read-only operations, but can also be used in online transaction processing (OLTP) systems.

For more details about when to use the compression clause, see the *Oracle Database Administrator's Guide*.

Tablespaces Created by Default: Overview

- EXAMPLE (optional)
- SYSAUX
- SYSTEM
- TEMP
- UNDOTBS1
- USERS



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.



The following tablespaces are created by default when you create an Oracle database:

- **SYSAUX**: This is an auxiliary tablespace to the SYSTEM tablespace. Some components and products that used the SYSTEM tablespace or their own tablespaces in earlier releases of Oracle Database now use the SYSAUX tablespace.
- **SYSTEM**: The SYSTEM tablespace is used by the Oracle server to manage the database. It contains the data dictionary and tables that contain administrative information about the database. These are all contained in the SYS schema and can be accessed only by the SYS user or other administrative users with the required privilege.
- **TEMP**: Your temporary tablespace is used when you execute a SQL statement that requires the creation of temporary segments (such as a large sort or the creation of an index). Just as each user is assigned a default tablespace for storing created data objects, each user is assigned a temporary tablespace. The best practice is to define a default temporary tablespace for the database, which is assigned to all newly created users unless otherwise specified. In the preconfigured database, the TEMP tablespace is specified as the default temporary tablespace. This means that if no temporary tablespace is specified when the user account is created, Oracle Database assigns this tablespace to the user.

- **UNDOTBS1:** This is the undo tablespace used by the database server to store undo information. If a database uses Automatic Undo Management, then it can only use a single undo tablespace at any given time. This tablespace is created at database creation time.
- **USERS:** This tablespace is used to store user objects and data. If no default tablespace is specified when a user is created, then the **USERS** tablespace is the default tablespace for all objects created by that user. For the **SYS** and **SYSTEM** users, the default permanent tablespace is **SYSTEM**.

The **EXAMPLE** tablespace contains the sample schemas that can be installed when you create the database. The sample schemas provide a common platform for examples. Oracle documentation and courseware contain examples based on the sample schemas.

Note: To simplify administration, it is common to have a tablespace for indexes alone.

Altering a Tablespace



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After you create a tablespace, you can alter it in several ways as the needs of your system change.

Changing the status: A tablespace can be in one of three different statuses or states. Any of the following three states may not be available because their availability depends on the type of tablespace.

- **Read Write:** The tablespace is online and can be read from and written to.
- **Read Only:** Specify read-only to place the tablespace in transition read-only mode. In this state, existing transactions can be completed (committed or rolled back), but no further data manipulation language (DML) operations are allowed on the objects in the tablespace. The tablespace is online while in the read-only state. You cannot make the SYSTEM or SYSAUX tablespaces read-only.

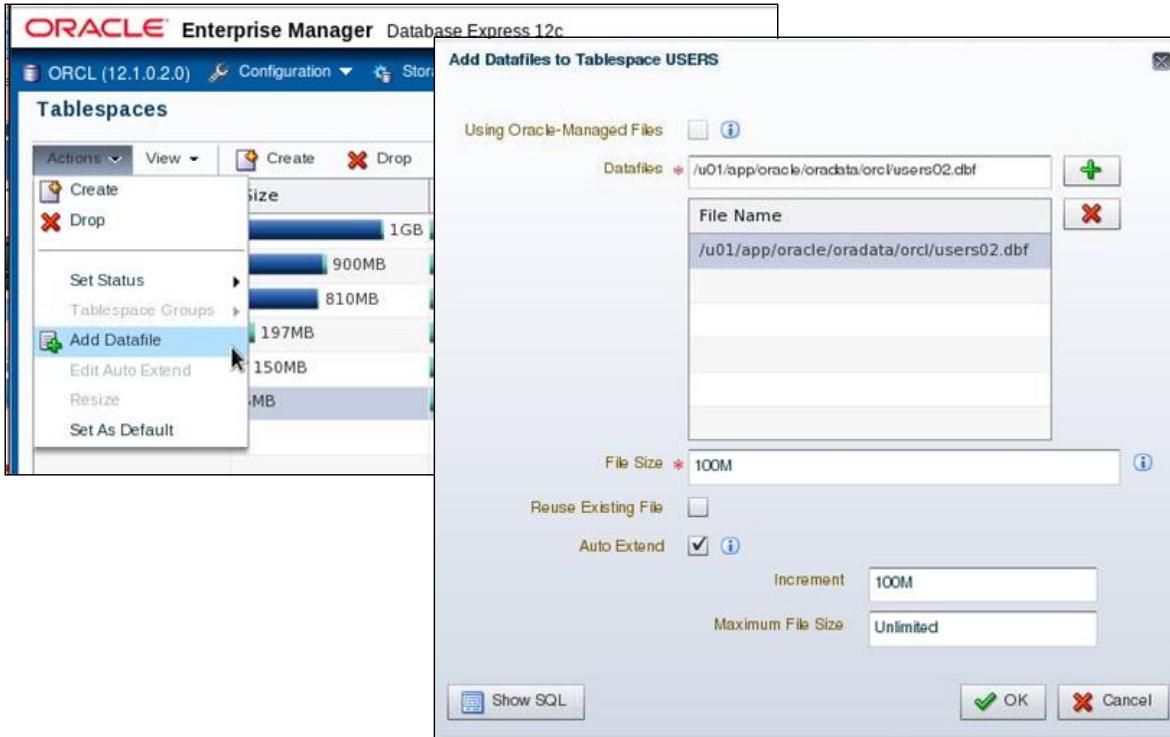
Note: The undo and temporary tablespaces cannot be made read-only.

- **Offline:** You can take an online tablespace offline so that this portion of the database is temporarily unavailable for general use. The rest of the database is open and available for users to access data. When you take it offline, you can use the following options:
 - **Normal:** A tablespace can be taken offline normally if no error conditions exist for any of the data files of the tablespace. Oracle Database ensures that all data is written to disk by taking a checkpoint for all data files of the tablespace as it takes them offline.

- **Temporary:** A tablespace can be taken offline temporarily even if there are error conditions for one or more files of the tablespace. Oracle Database takes the data files (which are not already offline) offline, performing checkpointing on them as it does so. If no files are offline, but you use the Temporary clause, media recovery is not required to bring the tablespace back online. However, if one or more files of the tablespace are offline because of write errors, and you take the tablespace offline temporarily, the tablespace requires recovery before you can bring it back online.
- **Immediate:** A tablespace can be taken offline immediately without Oracle Database taking a checkpoint on any of the data files. When you specify Immediate, media recovery for the tablespace is required before the tablespace can be brought online. You cannot take a tablespace offline immediately if the database is running in NOARCHIVELOG mode.

Note: System tablespaces may not be taken offline.

Adding a Data File to a Tablespace



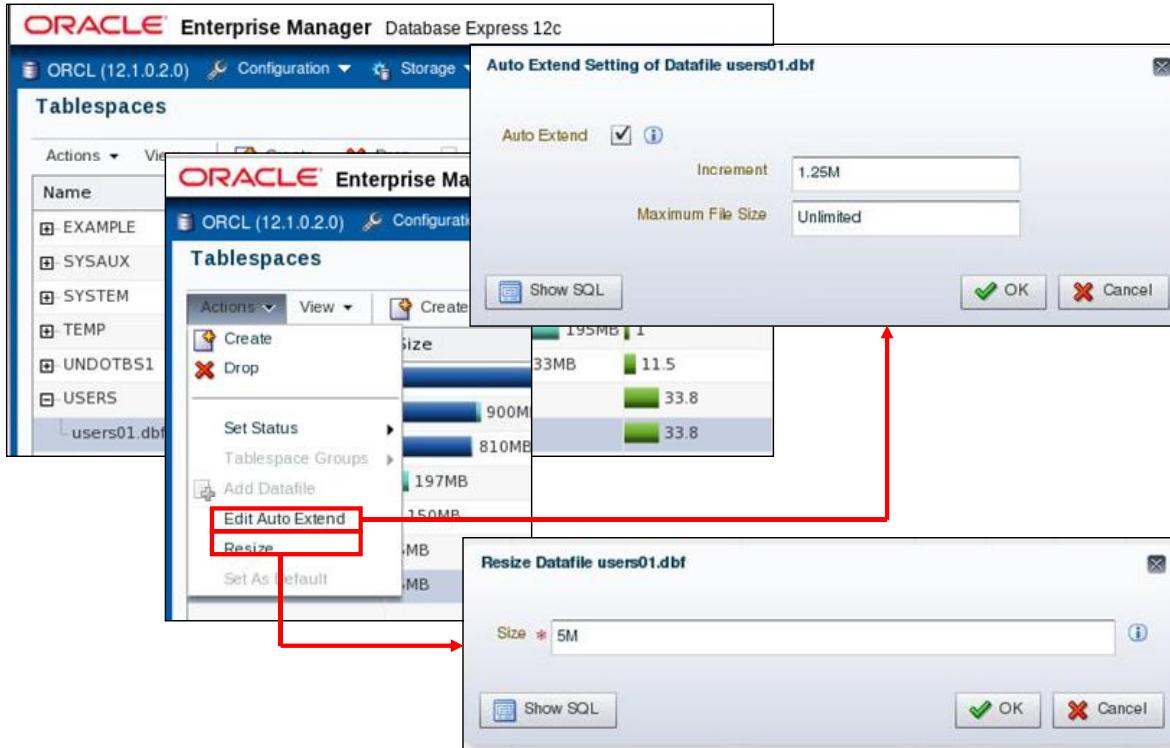
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Changing the size: You can add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file. To add a new data file to the tablespace, click Add. Then enter the information about the data file on the Add Datafile page.

Note: You cannot add additional data files to bigfile tablespaces.

Making Changes to a Data File



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can change the settings of a data file by expanding the tablespace on the Tablespaces page, selecting the data file, and selecting the appropriate task in the Actions menu.

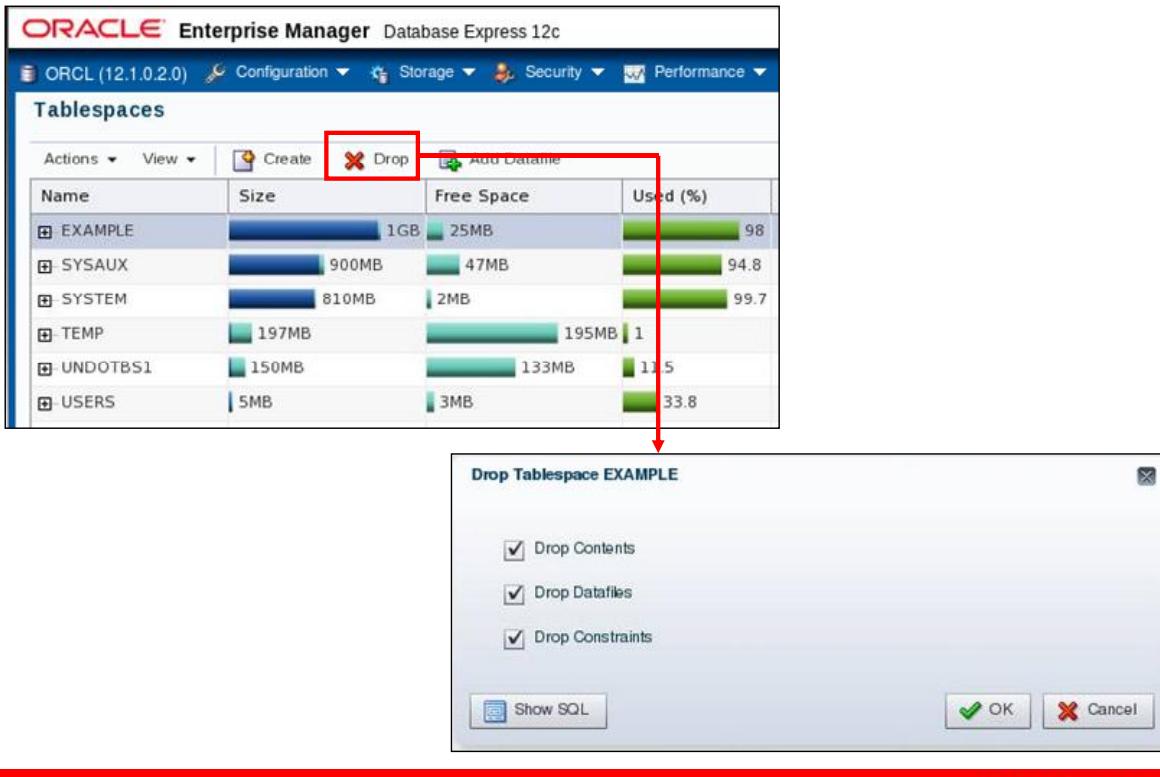
To take a data file offline, select “Take Offline” from the “Set Status” menu.

To change the size of an existing data file:

1. Expand the tablespace on the Tablespaces page and select the data file.
2. Select Resize in the Actions menu.
3. Then, on the Edit Datafile page, you can change the size of the data file. You can make the tablespace either larger or smaller. However, you cannot make a data file smaller than the used space in the file; if you try to do so, you get the following error:

ORA-03297: file contains used data beyond requested RESIZE value

Dropping Tablespaces



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. You must have the `DROP TABLESPACE` system privilege to drop a tablespace.

When you drop a tablespace, the file pointers in the control file of the associated database are removed. If you are using Oracle Managed Files (OMF), the underlying operating system files are also removed. Otherwise, without OMF, you can optionally direct the Oracle server to delete the operating system files (data files) that constitute the dropped tablespace. If you do not direct the Oracle server to delete the data files at the same time that it deletes the tablespace, you must later use the appropriate commands of your operating system if you want them to be deleted.

You cannot drop a tablespace that contains active segments. For example, if a table in the tablespace is currently being used, or if the tablespace contains undo data that is needed to roll back uncommitted transactions, you cannot drop the tablespace. It is best to take the tablespace offline before dropping it.

Viewing Tablespace Information

```
SQL> SELECT tablespace_name, status, contents, logging,
  2   extent_management, allocation_type,
  3   segment_space_management
  4  FROM dba_tablespaces;

TABLESPACE_NAME STATUS CONTENTS LOGGING EXTENT_MAN ALLOCATIO SEGMENT
----- ----- ----- ----- ----- ----- -----
SYSTEM      ONLINE PERMANENT LOGGING LOCAL    SYSTEM    MANUAL
SYSAUX      ONLINE PERMANENT LOGGING LOCAL    SYSTEM    AUTO
UNDOTBS1    ONLINE UNDO     LOGGING LOCAL    SYSTEM    MANUAL
TEMP        ONLINE TEMPORARY NOLOGGING LOCAL   UNIFORM   MANUAL
USERS       ONLINE PERMANENT LOGGING LOCAL    SYSTEM    AUTO
EXAMPLE     ONLINE PERMANENT NOLOGGING LOCAL    SYSTEM    AUTO

SQL> SELECT file_name, file_id, tablespace_name
  2  FROM dba_data_files;
FILE_NAME          FILE_ID TABLESPACE_NAME
----- ----- -----
/u01/app/oracle/oradata/orcl/system01.dbf           1  SYSTEM
/u01/app/oracle/oradata/orcl/sysaux01.dbf          3  SYSAUX
/u01/app/oracle/oradata/orcl/users01.dbf           6  USERS
/u01/app/oracle/oradata/orcl/example01.dbf         2  EXAMPLE
/u01/app/oracle/oradata/orcl/undotbs01.dbf         4  UNDOTBS1
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Tablespace and data file information can also be obtained by querying the following:

- **Tablespace information:**
 - DBA_TABLESPACES
 - V\$TABLESPACE
- **Data file information:**
 - DBA_DATA_FILES
 - V\$DATAFILE
- **Temp file information:**
 - DBA_TEMP_FILES
 - V\$TEMPFILE

Oracle Managed Files (OMF)

Specify file operations in terms of database objects rather than file names.

Parameter	Description
DB_CREATE_FILE_DEST	Defines the location of the default file system directory for data files and temporary files
DB_CREATE_ONLINE_LOG_DEST_n	Defines the location for redo log files and control file creation
DB_RECOVERY_FILE_DEST	Gives the default location for the fast recovery area

Example:

```
SQL> ALTER SYSTEM
  2  SET DB_CREATE_FILE_DEST='/u01/app/oracle/oradata';
SQL> CREATE TABLESPACE tbs_1;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Managed Files eliminate the need for you to directly manage the operating system files in an Oracle database. You specify operations in terms of database objects rather than file names. The database internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Redo log files
- Control files
- Archived logs
- Block change tracking files
- Flashback logs
- RMAN backups

A database can have a mixture of Oracle-managed and Oracle-unmanaged files. The file system directory specified by either of these parameters must already exist; the database does not create it. The directory must also have permissions for the database to create the files in it.

The example shows that after DB_CREATE_FILE_DEST is set, the DATAFILE clause can be omitted from a CREATE TABLESPACE statement. The data file is created in the location specified by DB_CREATE_FILE_DEST. When you create a tablespace as shown, default values are assigned to all parameters.

Oracle-managed files have a specific naming format. For example, on Linux- and UNIX-based systems the following format is used:

```
<destination_prefix>/o1_mf_%t_%u_.dbf
```

Do not rename an Oracle-managed file. The database identifies an Oracle-managed file based on its name. If you rename the file, the database is no longer able to recognize it as an Oracle-managed file and will not manage the file accordingly.

The following example sets the default location for data file creations to /u01/oradata and then creates a tablespace tbs_1 with a data file in that location.

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/u01/oradata';
SQL> CREATE TABLESPACE tbs_1;
```

By default, Oracle-managed data files, including those for the SYSTEM and SYSAUX tablespaces, are 100 MB and auto-extensible.

Note: By default, ASM uses OMF files, but if you specify an alias name for an ASM data file at tablespace creation time or when adding an ASM data file to an existing tablespace, then that file will not be OMF.

Quiz

A database can have a mixture of Oracle-managed and unmanaged files.

- a. True
- b. False



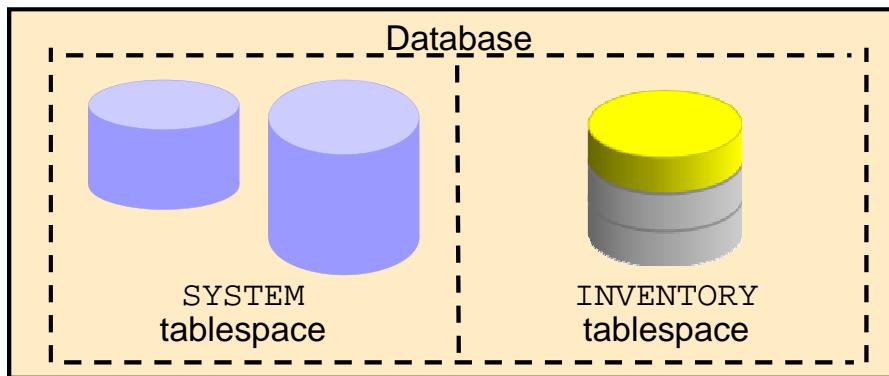
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Enlarging the Database

You can enlarge the database in the following ways:

- Create a new tablespace.
- Add a data file to an existing smallfile tablespace.
- Increase the size of a data file.
- Provide for the dynamic growth of a data file.

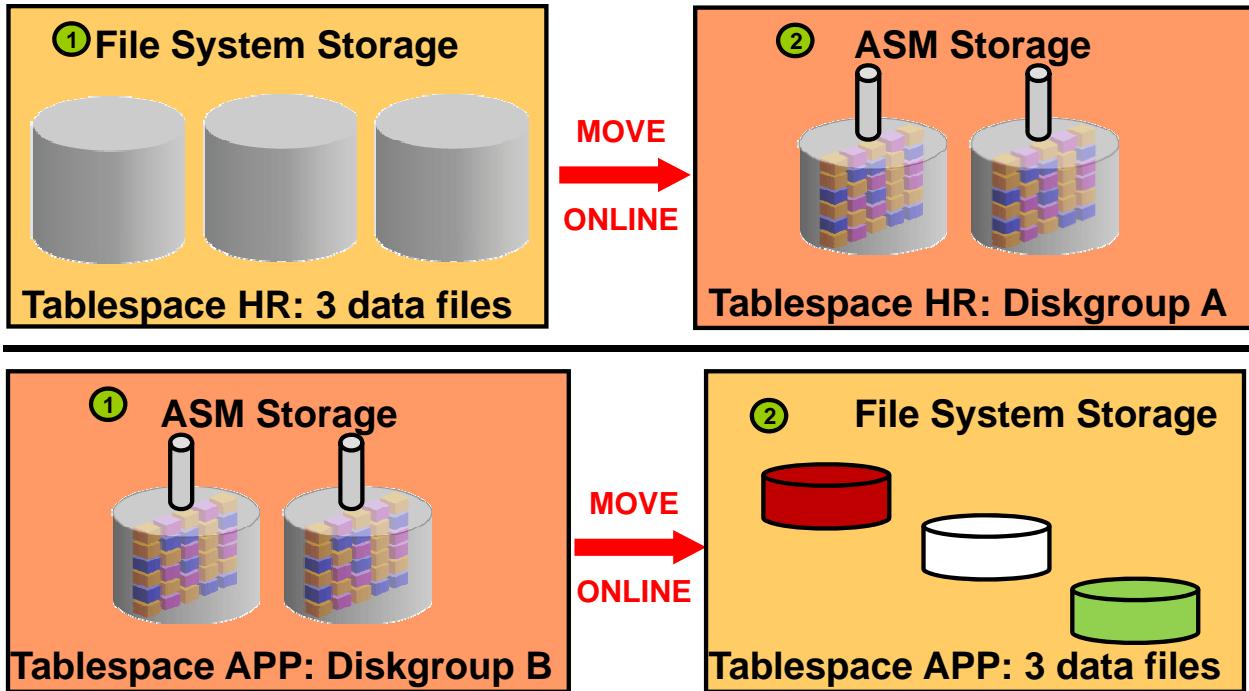


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

These activities can be performed with Enterprise Manager or with SQL statements. The size of the database can be described as the sum of all of its tablespaces.

Moving or Renaming an Online Data File



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can rename or move an online data file from one kind of storage system to another while the database is opened and accessing the file.

Queries and DML and DDL operations can be performed while the data file is being moved, including the following:

- SELECT statements against tables and partitions
- Creation of tables and indexes
- Rebuilding of indexes

Note: If objects are compressed while the data file is moved, the compression remains the same.

Moving or Renaming an Online Data File

- Relocating an online data file:

```
ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
      TO '/disk2/myexample01.dbf';
```

- Copying a data file from a file system to ASM:

```
ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
      TO '+DiskGroup2' KEEP;
```

- Renaming an online data file:

```
ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
      TO '/disk1/myexample01.dbf';
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You do not have to shut down the database or take the data file offline while you move a data file to another location, disk, or storage system.

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 option is specified, the existing file is overwritten.

If the KEEP clause is specified, the old file will be kept after the move operation. The KEEP clause is not allowed if the source file is an Oracle-managed file.

Use the V\$SESSION_LONGOPS view to display ongoing online move operations. Each ongoing operation has one row. The state transition of a successful online move operation is usually NORMAL to COPYING to SUCCESS and finally to NORMAL.

Summary

In this lesson, you should have learned how to:

- Describe the storage of table row data in blocks
- Create and manage tablespaces
- Obtain tablespace information



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Creating tablespaces
- Gathering information about tablespaces



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Oracle University and Error : You are not a Valid Partner use only

8

Managing Space

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Space Management: Overview

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

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



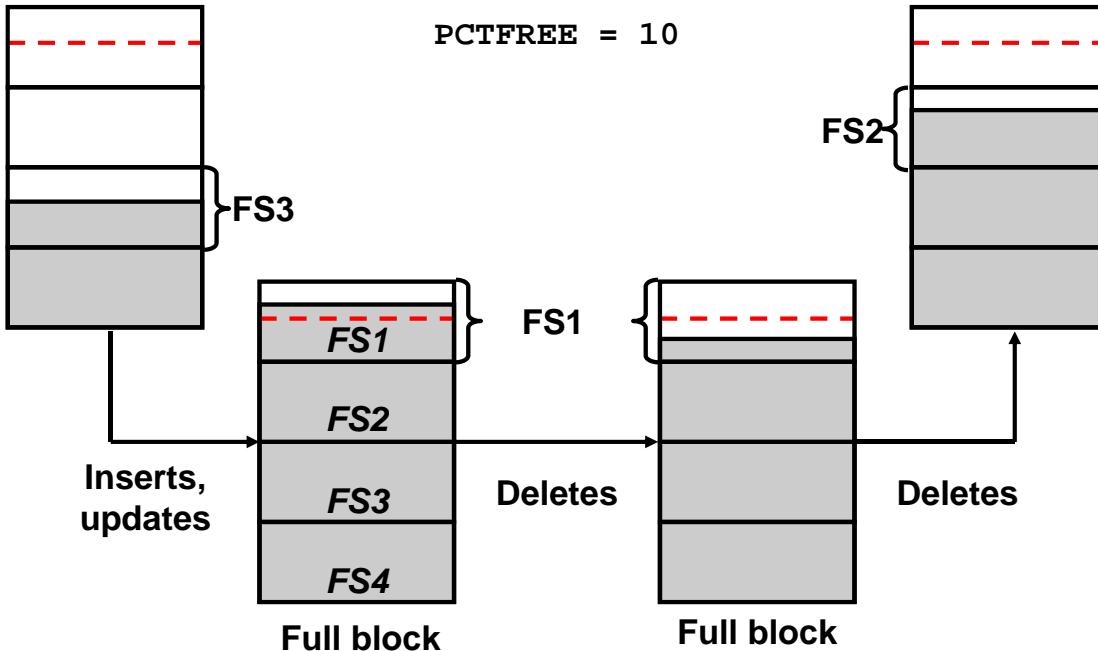
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

Block Space Management



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

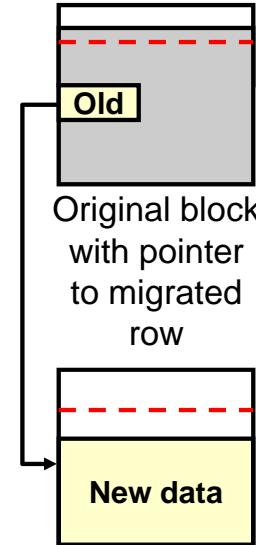
In the slide example, the block on the left is an FS3 block because it has between 50% and 75% free space. After some insert and update statements, `PCTFREE` is reached (the dashed line) and it is no longer possible to insert new rows in that block. The block is now considered as a “full” or FS1 block.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

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

Row Chaining and Migration

Example:

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

Quiz

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

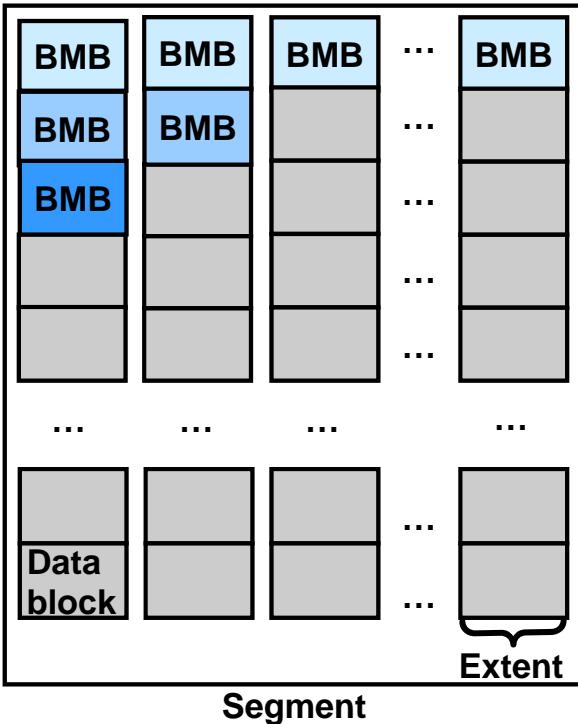
- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Free Space Management Within Segments



- Tracked by bitmaps in segments

Benefits:

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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

Benefits of using automatic space management include:

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

Types of Segments

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



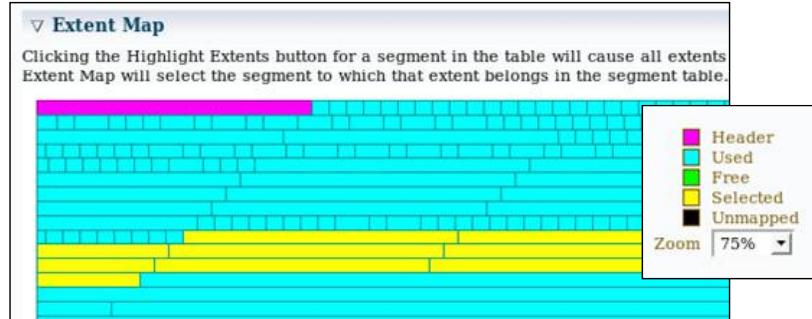
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **Table and cluster segments:** Each nonclustered table has a data segment. All table data is stored in the extents of the table segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- **Index segment:** Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
- **Undo segment:** Oracle Database maintains information to reverse changes made to the database. This information consists of records of the actions of transactions, collectively known as undo. Undo is stored in undo segments in an undo tablespace.
- **Temporary segment:** A temporary segment is created by the Oracle Database server when a SQL statement needs a temporary database area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.

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

Allocating Extents

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Two clauses affect the sizing of extents:

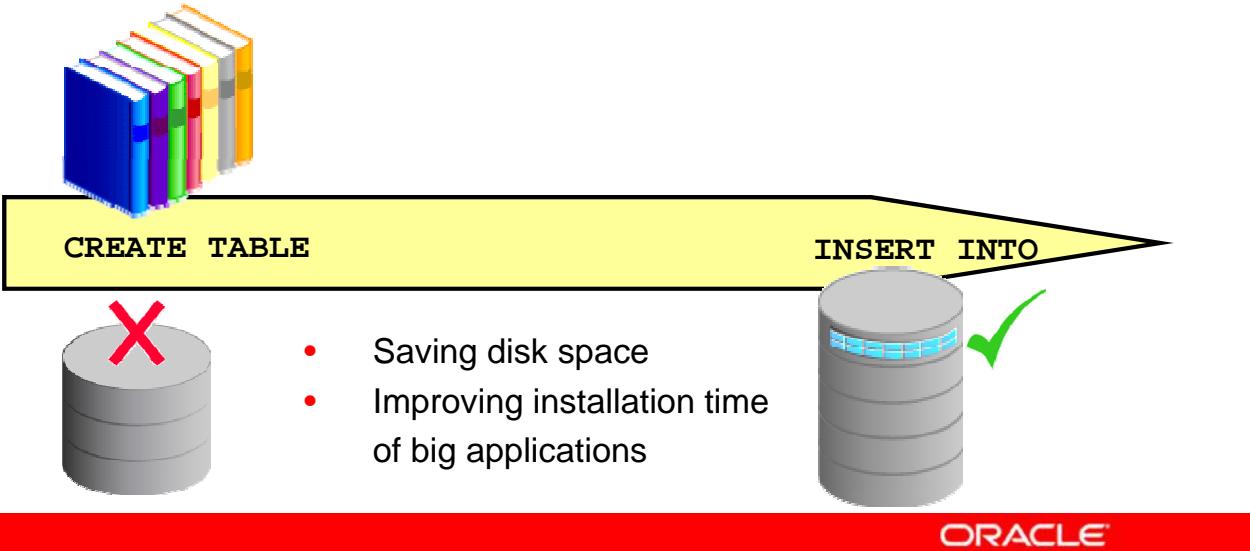
- With the **UNIFORM** clause, the database creates all extents of a uniform size that you specified (or a default size) for any objects created in the tablespace.
- With the **AUTOALLOCATE** clause, the database determines the extent-sizing policy for the tablespace.

To view the extent map in Enterprise Manager Cloud Control, choose Administration > Storage > Tablespaces. Select the tablespace and click View. Select Show Tablespace Contents in the Action menu and click Go. Expand Extent Map.

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

Understanding Deferred Segment Creation

- DEFERRED_SEGMENT_CREATION = TRUE is the default.
- Segment creation takes place as follows:
 1. Table creation > Data dictionary operation
 2. DML > Segment creation



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Advantages of this space allocation method:

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

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

Note: With this allocation method, it is essential that you do proper capacity planning so that the database has enough disk space to handle segment creation when tables are populated. For more details, see the *Oracle Database Administrator's Guide*.

Viewing Deferred Segment Information

```
SQL> SHOW PARAMETERS deferred_segment_creation
NAME                           TYPE        VALUE
-----
deferred_segment_creation      boolean     TRUE

SQL> CREATE TABLE seg_test(c number, d varchar2(500));
Table created.

SQL> SELECT segment_name FROM user_segments;
no rows selected
```

Inserting rows and creating segments:

```
SQL> INSERT INTO seg_test VALUES(1, 'aaaaaaaa');
1 row created.

SQL> SELECT segment_name FROM user_segments;
SEGMENT_NAME
-----
SEG_TEST
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This slide shows you how to check the DEFERRED_SEGMENT_CREATION parameter. Then a table is created without segments, which you can verify by querying the USER_SEGMENTS data dictionary view. After the insert of a row, you query this view again to see that the segment now exists.

You can also query the SEGMENT_CREATED column of the USER_TABLES, USER_INDEXES, or USER_LOBS views. For nonpartitioned tables, indexes, and LOBs, this column shows YES if the segment is created.

The SYS.SEG\$ data dictionary table stores the storage parameters that you specified during table or index creation.

Controlling Deferred Segment Creation

With the DEFERRED_SEGMENT_CREATION parameter in the:

- Initialization file
- ALTER SESSION command
- ALTER SYSTEM command

With the SEGMENT CREATION clause:

- IMMEDIATE
- DEFERRED (default)

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

Note: Indexes inherit table characteristics.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Segment creation can be controlled in two ways:

- With the DEFERRED_SEGMENT_CREATION initialization parameter set to TRUE or FALSE. This parameter can be set in the initialization file. You can also control it via the ALTER SESSION or ALTER SYSTEM commands.

Examples:

```
ALTER SESSION SET DEFERRED_SEGMENT_CREATION = TRUE;
ALTER SYSTEM SET DEFERRED_SEGMENT_CREATION = FALSE;
```

- With the SEGMENT CREATION clause of the CREATE TABLE command:
 - SEGMENT CREATION DEFERRED: If specified, segment creation is deferred until the first row is inserted into the table. This is the default behavior for Oracle Database 11g Release 2 and later releases.
 - SEGMENT CREATION IMMEDIATE: If specified, segments are materialized during table creation. This is the default behavior in Oracle databases before Oracle Database 11g Release 2.

This clause takes precedence over the DEFERRED_SEGMENT_CREATION parameter.

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

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

Restrictions and Exceptions

- Segment creation on demand is:
 - Only for nonpartitioned tables and indexes
 - Not for IOTs, clustered tables, or other special tables
 - Not for tables in dictionary-managed tablespaces
- If you were to migrate a table without segments from a locally managed to a dictionary-managed tablespace, you must drop and re-create it.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Deferred segment creation is restricted to nonpartitioned tables and nonpartitioned indexes.

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

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

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

Additional Automatic Functionality

Without user intervention:

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

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

- Removing any allocated space for an index:

```
ALTER INDEX test_i UNUSABLE;
```

- Creating the segment for an index:

```
ALTER INDEX test_i REBUILD;
```

```
SELECT segment_name, partition_name, segment_type  
FROM user_segments  
WHERE segment_name like '%DEMO';
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Example: If you have a DEMO table with three partitions and a local index, you see three table and three index segments when executing the query, which is shown in the slide.

If you execute the same query after you move one table partition to a new tablespace, you see three table segments and only two index segments because the unusable one is automatically deleted.

Quiz

Which of the following statements are true?

- a. Deferred segment creation is always enabled. You cannot control it.
- b. You can control the deferred segment creation with the SEGMENT CREATION clause of the CREATE TABLE command.
- c. Segment creation on demand is available for all types of tables, including those owned by the SYS user.
- d. Segment creation on demand is available for nonpartitioned tables.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

Table Compression: Overview

Reducing storage costs by compressing all data:

- Basic compression for direct-path insert operations: **10x**
- Advanced row compression for all DML operations: **2–4x**

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Database supports three methods of table compression:

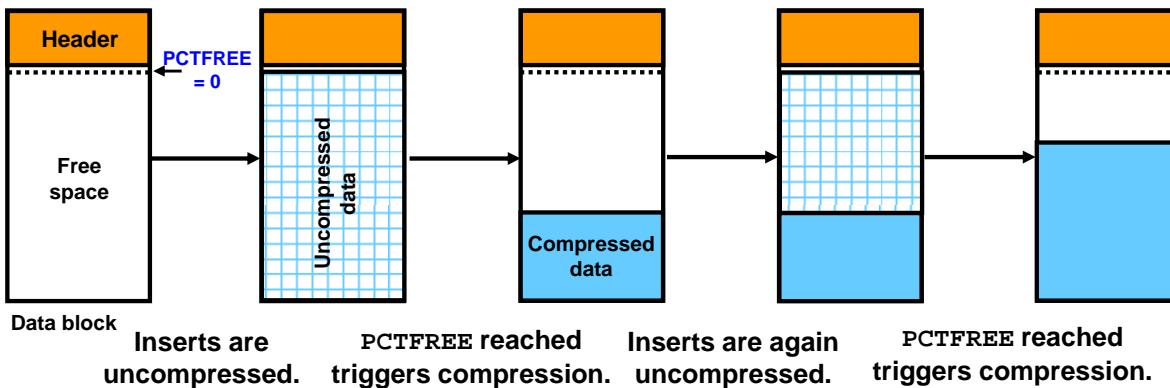
- Basic table compression
- Advanced row compression
- Hybrid columnar compression (with Exadata)

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

- The `table_compression` clause is valid only for heap-organized tables. The `COMPRESS` keyword enables table compression. The `NOCOMPRESS` keyword disables table compression. `NOCOMPRESS` is the default.
- With basic compression, the Oracle Database server compresses data at the time of performing bulk load using operations such as direct loads or `CREATE TABLE AS SELECT`.
- With `ROW STORE COMPRESS ADVANCED`, the Oracle Database server compresses data during all DML operations on the table.

Compression for Direct-Path Insert Operations

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

With `COMPRESS` or `COMPRESS BASIC`, you enable basic table compression.

- The Oracle Database server attempts to compress data during the following direct-path insert operations when it is productive to do so:
 - Direct-path SQL*Loader
 - `CREATE TABLE AS SELECT` statements
 - Parallel `INSERT` statements
 - `INSERT` statements with an `APPEND` hint
- The original import utility (`imp`) does not support direct-path `INSERT`, and therefore cannot import data in a compressed format.
- In earlier releases, this type of compression was called DSS table compression and was enabled using `COMPRESS FOR DIRECT_LOAD OPERATIONS`. This syntax has been deprecated.
- Compression eliminates holes created due to deletions and maximizes contiguous free space in blocks.

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

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

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

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

Advanced Row Compression for DML Operations

- Is enabled with

```
CREATE TABLE ... ROW STORE COMPRESS ADVANCED  
...;
```

- Is recommended for active OLTP environments

	Y	Y	Y								
G	Y		G								
G		Y	Y	Y	G						

Uncompressed block

G	Y										
	Y		Y								
G		Y		G							
G			Y	Y	G						

OLTP compression with symbol table at the beginning of the block



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

With `ROW STORE COMPRESS ADVANCED`, you enable advanced row compression.

- The Oracle database compresses data during all DML operations on the table. This form of compression is recommended for active OLTP environments.
- In earlier releases, OLTP table compression was enabled with `COMPRESS FOR ALL OPERATIONS` and `COMPRESS FOR OLTP`. This syntax has been deprecated.

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

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

Specifying Table Compression

You can specify table compression for:

- An entire heap-organized table
- A partitioned table (Each partition can have a different type or level of compression.)
- The storage of a nested table

You cannot :

- Specify basic and advanced row compression on tables with more than 255 columns
- Drop a column if a table is compressed for direct-loads, but you can drop it if the table is advance row compressed



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can specify table compression:

- For an entire heap-organized table (in the *physical_properties* clause of *relational_table* or *object_table*)
- For partitioned tables (Each partition can have a different type or level of compression.)
- For the storage of a nested table (in the *nested_table_col_properties* clause)

Table compression has the following restrictions:

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

Using the Compression Advisor

The Compression Advisor:

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Using the DBMS_COMPRESSION Package

To determine optimal compression ratios:

```
BEGIN
  DBMS_COMPRESSION.GET_COMPRESSION_RATIO ('USERS','SH','SALES',
    NULL,DBMS_COMPRESSION.COMP_FOR_OLTP, blkcnt_cmp, blkcnt_ncmp,
    rowcnt_cmp, rowcnt_ncmp, comp_ratio, comptype);
  DBMS_OUTPUT.PUT_LINE('Blk count compressed = ' || blkcnt_cmp);
  DBMS_OUTPUT.PUT_LINE('Blk count uncompressed = ' || blkcnt_ncmp);
  DBMS_OUTPUT.PUT_LINE('Row count per block compressed = ' || rowcnt_cmp);
  DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed = ' || rowcnt_ncmp);
  DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype);
  DBMS_OUTPUT.PUT_LINE('Compression ratio = '||comp_ratio||' to 1');
END;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

To determine the compression ratio, the DBMS_COMPRESSION package has the following subprograms:

- The GET_COMPRESSION_RATIO procedure gives you the possible compression ratio for an uncompressed table.
- The GET_COMPRESSION_TYPE function returns the compression type for a given row.

For more details, see the *Oracle Database PL/SQL Packages and Types Reference*.

Proactive Tablespace Monitoring

The screenshot displays two panels of the Oracle Enterprise Manager Cloud Control interface, both titled "Edit Tablespace: EXAMPLE".

Top Panel (Storage Tab): This panel shows the "Extent Allocation" section with "Allocation Type" set to "Automatic". It also includes "Segment Space Management" (Type: Automatic) and "Compression Options". The compression section notes: "Enable data segment compression to reduce disk and cache usage. This can be used in both OLTP and OLAP environments." It lists four compression types: "No Compression" (selected), "Basic Compression", "CLTP Compression", and "Warehouse Compression".

Bottom Panel (Thresholds Tab): This panel shows the "Thresholds" tab selected. It displays disk space usage metrics: "Available Space (MB)" (32767.98), "Space Used (%)" (3.72), "Space Used (MB)" (1219.06), and "Available Free Space (MB)" (31548.92). A note at the bottom states: "To view or modify Tablespace Space Used (%) or Free Space (MB) thresholds, navigate to the Database Monitoring Metric and Collection Settings page." Below this is a link to "Metric and Collection Settings".

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Tablespace disk space usage is proactively managed by the database in the following ways:

- Through the use of database alerts, you are informed when a tablespace runs low on available disk space as well as when particular segments are running out of space. You can then provide the tablespace with more disk space, thus avoiding out-of-space conditions.
- Information gathered is stored in the Automatic Workload Repository (AWR) and is used to perform growth trend analysis and capacity planning of the database.

To view and modify tablespace information in Enterprise Manager Cloud Control, select Administration > Storage > Tablespaces. Select the tablespace of your choice and click Edit.

Thresholds and Resolving Space Problems



Locally managed tablespace

Resolve space problem by:

- Adding or resizing data file
- Setting AUTOEXTEND ON
- Shrinking objects
- Reducing UNDO_RETENTION
- Checking for long-running queries in temporary tablespaces

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

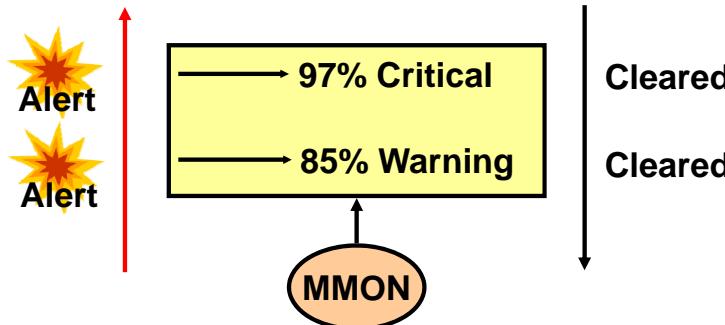
Tablespace thresholds are defined either as full or as available space in the tablespace. Critical and warning thresholds are the two thresholds that apply to a tablespace. The DBMS_SERVER_ALERT package contains procedures to set and get the threshold values. When the tablespace limits are reached, an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

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

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

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

Monitoring Tablespace Space Usage



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

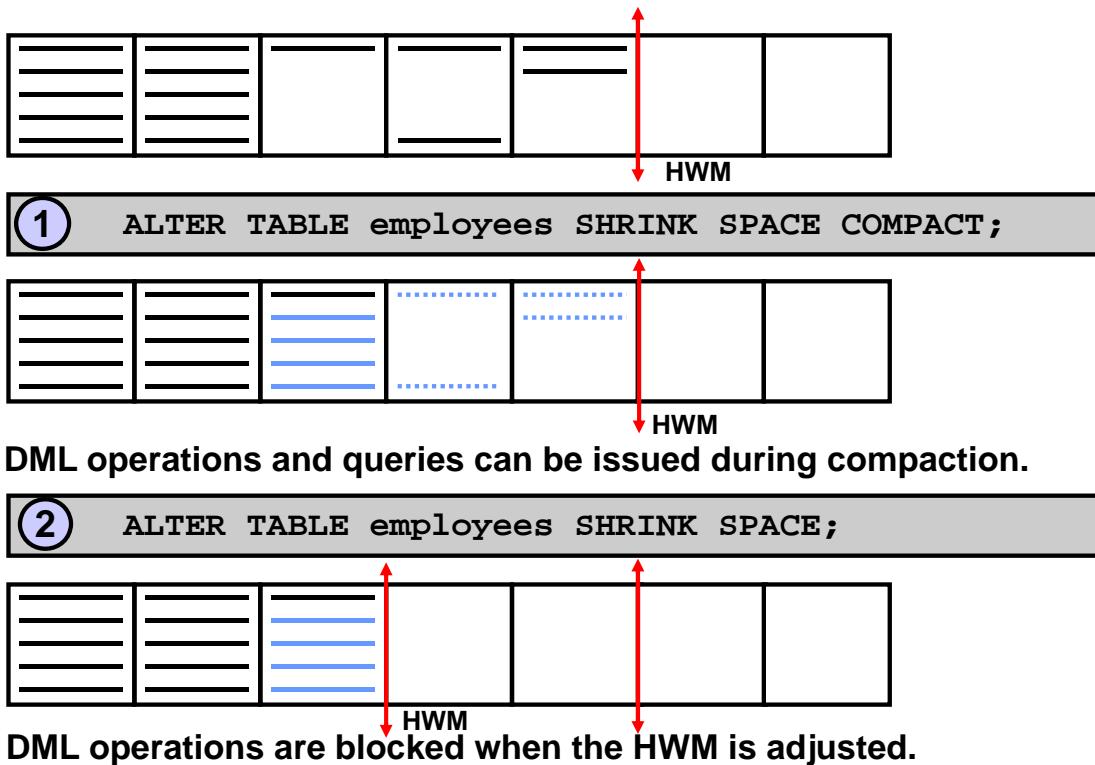
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

- Alerts should not be flagged on tablespaces that are in read-only mode, or tablespaces that were taken offline, because there is not much to do for them.
- In temporary tablespaces, the threshold value has to be defined as a limit on the used space in the tablespace.
- For undo tablespaces, an extent is reusable if it does not contain active or unexpired undo. For the computation of threshold violation, the sum of active and unexpired extents is considered as used space.
- For tablespaces with auto-extensible files, the thresholds are computed according to the maximum file size you specified, or the maximum OS file size.

Shrinking Segments



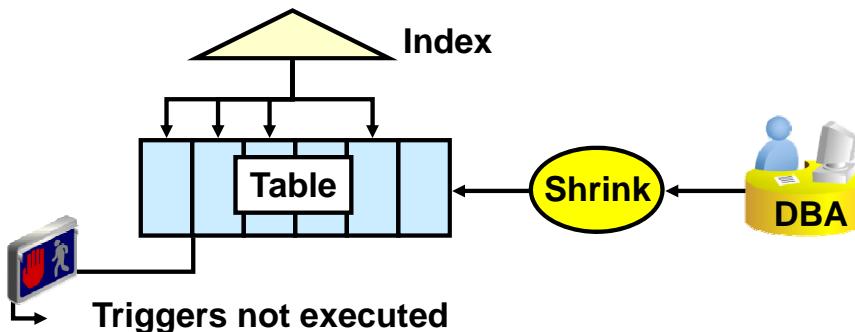
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide describes the two phases of a table shrink operation. Compaction is performed in the first phase. During this phase, rows are moved to the left part of the segment as much as possible. Internally, rows are moved by packets to avoid locking issues. After the rows have been moved, the second phase of the shrink operation is started. During this phase, the high-water mark (HWM) is adjusted and the unused space is released. The COMPACT clause is useful if you have long-running queries that might span the shrink operation and attempt to read from blocks that have been reclaimed. When you specify the SHRINK SPACE COMPACT clause, the progress of the shrink operation is saved in the bitmap blocks of the corresponding segment. This means that the next time a shrink operation is executed on the same segment, the Oracle Database server remembers what has been done already. You can then reissue the SHRINK SPACE clause without the COMPACT clause during off-peak hours to complete the second phase.

Results of Shrink Operation

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

Reclaiming Space Within ASSM Segments

- Online and in-place operation
- Applicable only to segments residing in ASSM tablespaces
- Candidate segment types:
 - Heap-organized tables and index-organized tables
 - Indexes
 - Partitions and subpartitions
 - Materialized views and materialized view logs



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

- You cannot execute a shrink operation on segments managed by free lists. Segments in automatic segment space-managed tablespaces can be shrunk. However, the following objects stored in ASSM tablespaces cannot be shrunk:
 - Tables in clusters
 - Tables with LONG columns
 - Tables with on-commit materialized views
 - Tables with ROWID-based materialized views
 - IOT mapping tables
 - Tables with function-based indexes
- ROW MOVEMENT must be enabled for heap-organized segments.

Using the Segment Advisor

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. At the top, there's a navigation bar with links for Enterprise, Targets, Favorites, History, and a search bar for 'Search Target Name'. On the right, it shows 'Logged in as dba1' and 'ADMIN'. Below the navigation is a breadcrumb trail: Oracle Database > Performance > Availability > Security > Schema > Administration. A progress bar at the top indicates steps: Scope, Objects, Schedule, Review. The main content area is titled 'Segment Advisor: Scope' for database 'orcl'. It shows 'Logged In As dba1' and buttons for 'Cancel', 'Step 1 of 4', and 'Next'. A yellow box highlights 'Automatic Segment Advisor Information': 'Beginning in Oracle Database 11, Oracle provides an Automatic Segment Advisor task which automatically detects segment issues.' To the left, there's a note: 'You can get advice on shrinking segments for individual schema objects or entire tablespaces.' with radio buttons for 'Tablespaces' (selected) and 'Schema Objects'. To the right, there's an 'Overview' section: 'The segment advisor determines whether objects have unused space that can be released, taking estimated future space requirements into consideration. The estimated future space calculation is based on historical trends.' Below this is another set of 'Cancel', 'Step 1 of 4', and 'Next' buttons.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Segment Advisor identifies segments that have space available for reclamation. It performs its analysis by examining usage and growth statistics in the Automatic Workload Repository (AWR), and by sampling the data in the segment. It is configured to run automatically at regular intervals, and you can also run it on demand (manually). The regularly scheduled Segment Advisor run is known as the Automatic Segment Advisor.

After the recommendations are made, you can choose to implement the recommendations. The shrink advisor can be invoked at the segment or tablespace level.

Use Enterprise Manager Database Cloud Control to invoke the Segment Advisor. You can access the Segment Advisor from several places within Enterprise Manager Cloud Control:

- Advisor Home page
- Tablespaces page
- Schema object pages

Enterprise Manager Cloud Control provides the option to select various inputs and schedule a job that calls the Segment Advisor to get shrink advice. The Segment Advisor Wizard can be invoked with no context, in the context of a tablespace, or in the context of a schema object.

The Segment Advisor makes recommendation on the basis of sampled analysis, historical information, and future growth trends.

Automatic Segment Advisor

The Automatic Segment Advisor:

- Is started by a Scheduler job set to run during the default maintenance window:
 - Weeknights, Monday–Friday, from 10:00 PM to 2:00 AM
 - Saturday and Sunday, both windows start at 6:00 AM and last for 20 hours
- Examines database statistics, samples segment data, and then selects the following objects to analyze:
 - Tablespaces that have exceeded a critical or warning threshold
 - Segments that have the most activity
 - Segments that have the highest growth rate



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Automatic Segment Advisor is started by a Scheduler job that is configured to run during the default maintenance window. The default maintenance window is specified in the Scheduler, and is initially defined as follows:

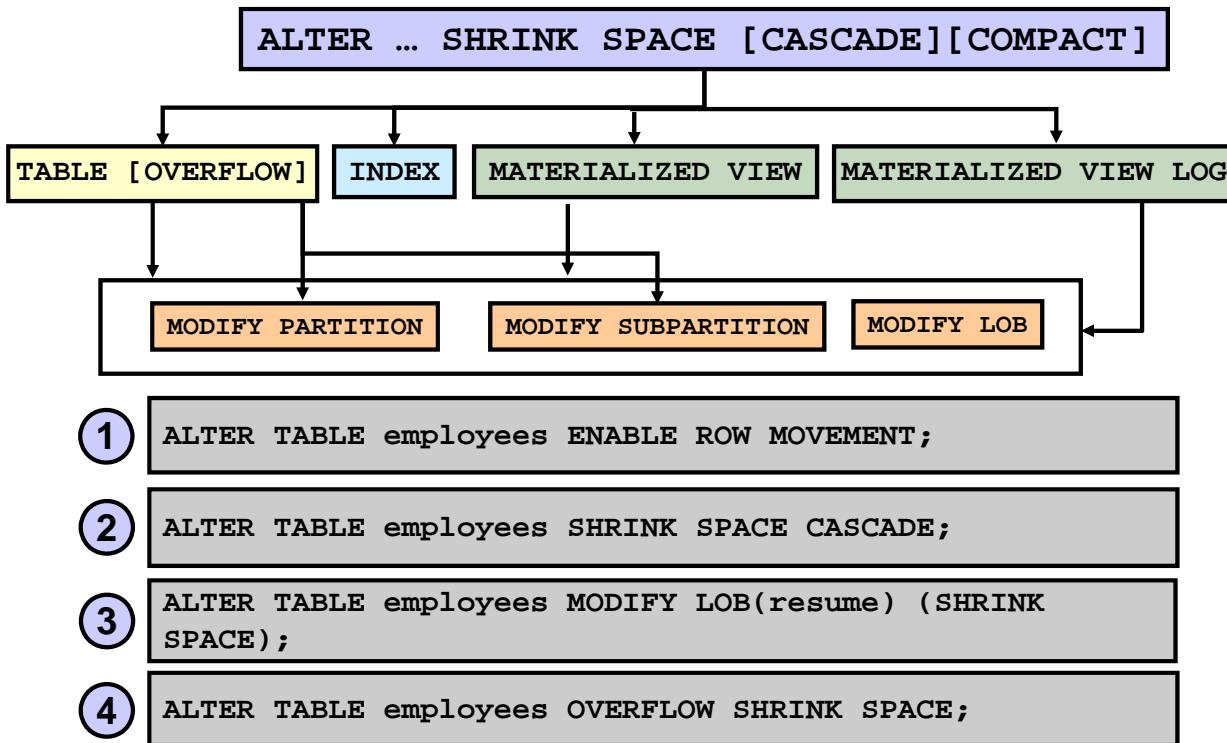
- Weeknights, Monday through Friday, from 10:00 PM to 2:00 AM (4 hours each night)
- Weekends, Saturday and Sunday morning at 6:00 AM and lasting for 20 hours each day.

The Automatic Segment Advisor does not analyze every database object. Instead, it examines database statistics, samples segment data, and then selects the following objects to analyze:

- Tablespaces that have exceeded a critical or warning space threshold
- Segments that have the most activity
- Segments that have the highest growth rate

If an object is selected for analysis but the maintenance window expires before the Segment Advisor can process the object, the object is included in the next Automatic Segment Advisor run. You cannot change the set of tablespaces and segments that the Automatic Segment Advisor selects for analysis. You can, however, enable or disable the Automatic Segment Advisor job, change the times during which the Automatic Segment Advisor is scheduled to run, or adjust Automatic Segment Advisor system resource utilization.

Shrinking Segments by Using SQL



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Because a shrink operation may cause ROWIDs to change in heap-organized segments, you must enable row movement on the corresponding segment before executing a shrink operation on that segment. Row movement by default is disabled at segment level. To enable row movement, the `ENABLE ROW MOVEMENT` clause of the `CREATE TABLE` or `ALTER TABLE` command is used. This is illustrated in the first example in the slide.

Use the `ALTER` command to invoke segment shrink on an object. The object's type can be one of the following: table (heap- or index-organized), partition, subpartition, LOB (data and index segment), index, materialized view, or materialized view log.

Use the `SHRINK SPACE` clause to shrink space in a segment. If `CASCADE` is specified, the shrink behavior is cascaded to all the dependent segments that support a shrink operation, except materialized views, LOB indexes, and IOT (index-organized tables) mapping tables. The `SHRINK SPACE` clause is illustrated in the second example.

In an index segment, the shrink operation coalesces the index before compacting the data. Example 3 shows a command that shrinks an LOB segment, given that the `RESUME` column is a `CLOB`.

Example 4 shows a command that shrinks an IOT overflow segment belonging to the `EMPLOYEES` table.

Note: For more information, refer to the *Oracle Database SQL Reference* guide.

Shrinking Segments by Using Enterprise Manager

Selection Mode: Single

Select	Schema	Table Name	Tablespace	Partitioned
<input type="radio"/>	HR	COUNTRIES	EXAMPLE	NO
<input type="radio"/>	HR	DEPARTMENTS	EXAMPLE	NO
<input checked="" type="radio"/>	HR	EMPLOYEES	EXAMPLE	NO
<input type="radio"/>	HR	JOB_H		
<input type="radio"/>	HR	LOCATI		
<input type="radio"/>	HR	REGION		

Shrink Options

- Compact Segments and Release Space
This will first compact the segments and then release the recovered space to the tablespace. During the short space release affected.
- Compact Segments
Compacting will compact segment data without releasing the recovered space. After compacting the data, the recovered

Segment Selection

- Shrink HR.EMPLOYEES Only
- Shrink HR.EMPLOYEES and All Dependent Segments

Dependent Segments

Schema	Segment Name	Type
HR	EMPLOYEES	TABLE
HR	EMP_EMP_ID_PK	INDEX
HR	EMP_DEPARTMENT_IX	INDEX
HR	EMP_JOB_IX	INDEX
HR	EMP_EMAIL_UK	INDEX
HR	EMP_NAME_IX	INDEX
HR	EMP_MANAGER_IX	INDEX

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can manually shrink individual segments associated with specific database objects by using Enterprise Manager Cloud Control. On the Tables page, select your table, and then select Shrink Segment in the Actions drop-down list. Then click the Go button. This brings you to the Shrink Segment page, where you can choose the dependent segments to shrink. You have the opportunity to compact only or to compact and release the space. You can also choose the CASCADE option.

When done, click the Continue link. This submits the shrink statements as a scheduled job.

Managing Resumable Space Allocation

A resumable statement:

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

Using Resumable Space Allocation

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

```
ALTER SESSION ENABLE RESUMABLE;
INSERT INTO sales_new SELECT * FROM sh.sales;
ALTER SESSION DISABLE RESUMABLE;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

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

Example:

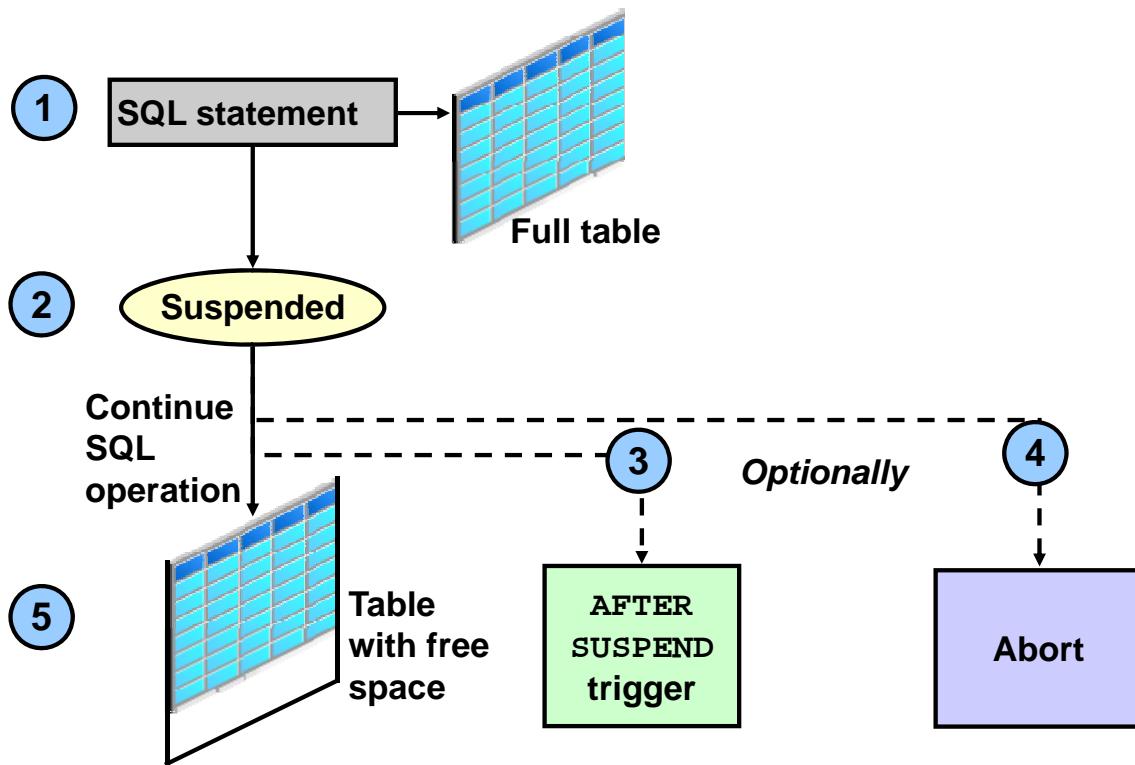
```
SELECT name, sql_text FROM user_resumable;
```

NAME	SQL_TEXT
multitab insert	INSERT INTO oldsales SELECT * FROM sh.sales;

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

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

Resuming Suspended Statements



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Example

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

Detecting a Suspended Statement

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

Possible Actions During Suspension

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

Within the trigger code, you can use the USER_RESUMABLE or DBA_RESUMABLE views, or the DBMS_RESUMABLE.SPACE_ERROR_INFO function to get information about the resumable statements.

When a resumable statement is suspended:

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

Ending a Suspended Statement

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

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

What Operations Are Resumable?

The following operations are resumable:

- Queries: SELECT statements that run out of temporary space (for sort areas)
- DML: INSERT, UPDATE, and DELETE statements
- The following DDL statements:
 - CREATE TABLE ... AS SELECT
 - CREATE INDEX
 - ALTER INDEX ... REBUILD
 - ALTER TABLE ... MOVE PARTITION
 - ALTER TABLE ... SPLIT PARTITION
 - ALTER INDEX ... REBUILD PARTITION
 - ALTER INDEX ... SPLIT PARTITION
 - CREATE MATERIALIZED VIEW



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following operations are resumable:

- **Queries:** SELECT statements that run out of temporary space (for sort areas) are candidates for resumable execution. When using OCI, the OCISqlExecute() and OCISqlFetch() calls are candidates.
- **DML:** INSERT, UPDATE, and DELETE statements are candidates. The interface used to execute them does not matter; it can be OCI, SQLJ, PL/SQL, or another interface. Also, INSERT INTO...SELECT from external tables can be resumable.
- **DDL:** The following statements are candidates for resumable execution:
 - CREATE TABLE ... AS SELECT
 - CREATE INDEX
 - ALTER INDEX ... REBUILD
 - ALTER TABLE ... MOVE PARTITION
 - ALTER TABLE ... SPLIT PARTITION
 - ALTER INDEX ... REBUILD PARTITION
 - ALTER INDEX ... SPLIT PARTITION
 - CREATE MATERIALIZED VIEW

Quiz

Select the true statements about space management:

- a. Segment creation in Oracle Database 12c is deferred for all tables. There are no exceptions.
- b. All UNUSABLE indexes and index partitions are created without a segment.
- c. Shrinking segment space is a nonresumable operation.
- d. You can set thresholds by tablespace.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

Summary

In this lesson, you should have learned how to:

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Using threshold alerts to proactively manage tablespaces
- Using the Segment Advisor to shrink space
- Viewing alerts and alert history in SQL*Plus and Enterprise Manager



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

9

Managing Undo Data

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

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

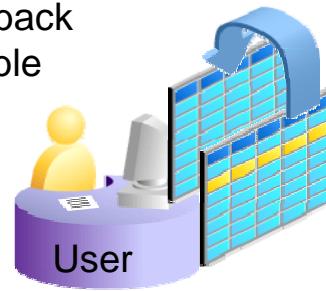


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Undo Data: Overview

Undo data is:

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

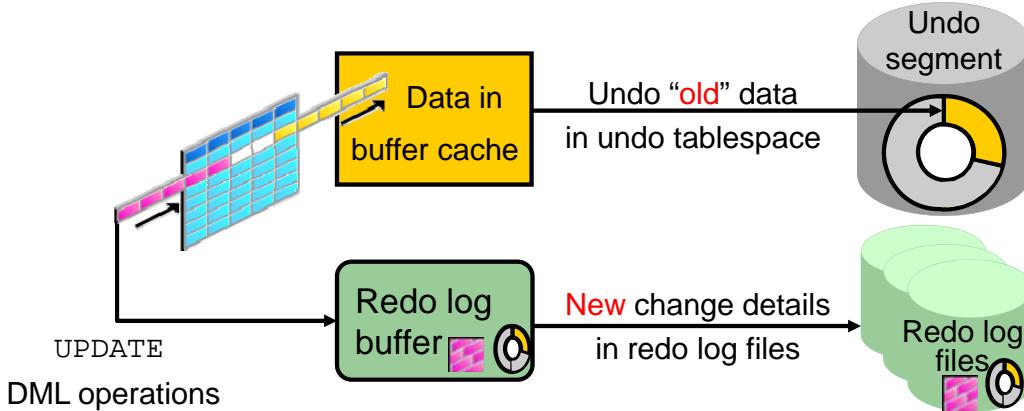
Undo information is retained for all transactions, at least until the transaction is ended by one of the following:

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

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

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

Transactions and Undo Data



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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Undo segments are specialized segments that are automatically created by the database server as needed to support transactions. Like all segments, undo segments are made up of extents, which, in turn, consist of data blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions.

Transactions fill extents in their undo segments until a transaction is completed or all space is consumed. If an extent fills up and more space is needed, the transaction acquires that space from the next extent in the segment. After all extents have been consumed, the transaction either wraps around back into the first extent or requests a new extent to be allocated to the undo segment.

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

Storing Undo Information

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

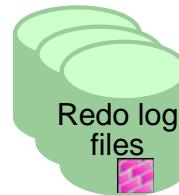
Undo segments are automatically created and always owned by `SYS`. Because the undo segments act as a circular buffer, each segment has a minimum of two extents. The default maximum number of extents depends on the database block size but is very high (32,765 for an 8 KB block size).

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

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

Comparing Undo Data and Redo Data

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

Managing Undo

Automatic undo management:

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

DBA tasks in support of Flashback operations:

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

Configuring Undo Retention

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

Categories of Undo

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



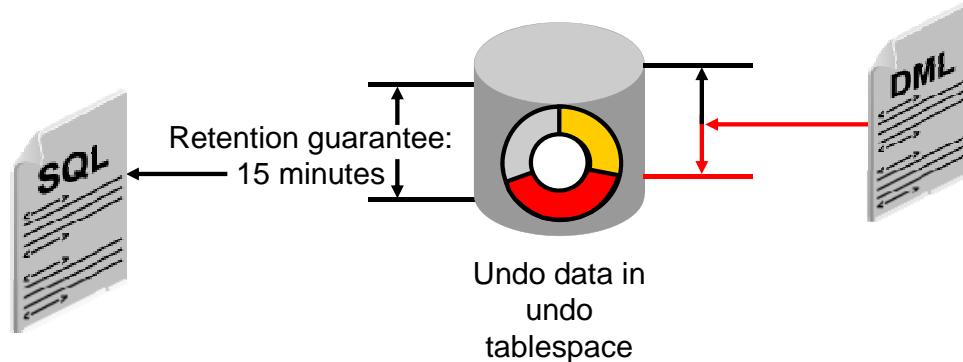
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Undo information is divided into three categories:

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

Guaranteeing Undo Retention

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



SELECT statements running 15 minutes or less are always satisfied.

A transaction will fail if it generates more undo than there is space.

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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

RETENTION GUARANTEE is a tablespace attribute rather than an initialization parameter. This attribute can be changed only with SQL command-line statements. The syntax to change an undo tablespace to guarantee retention is:

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

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

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

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

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

Changing an Undo Tablespace to a Fixed Size

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

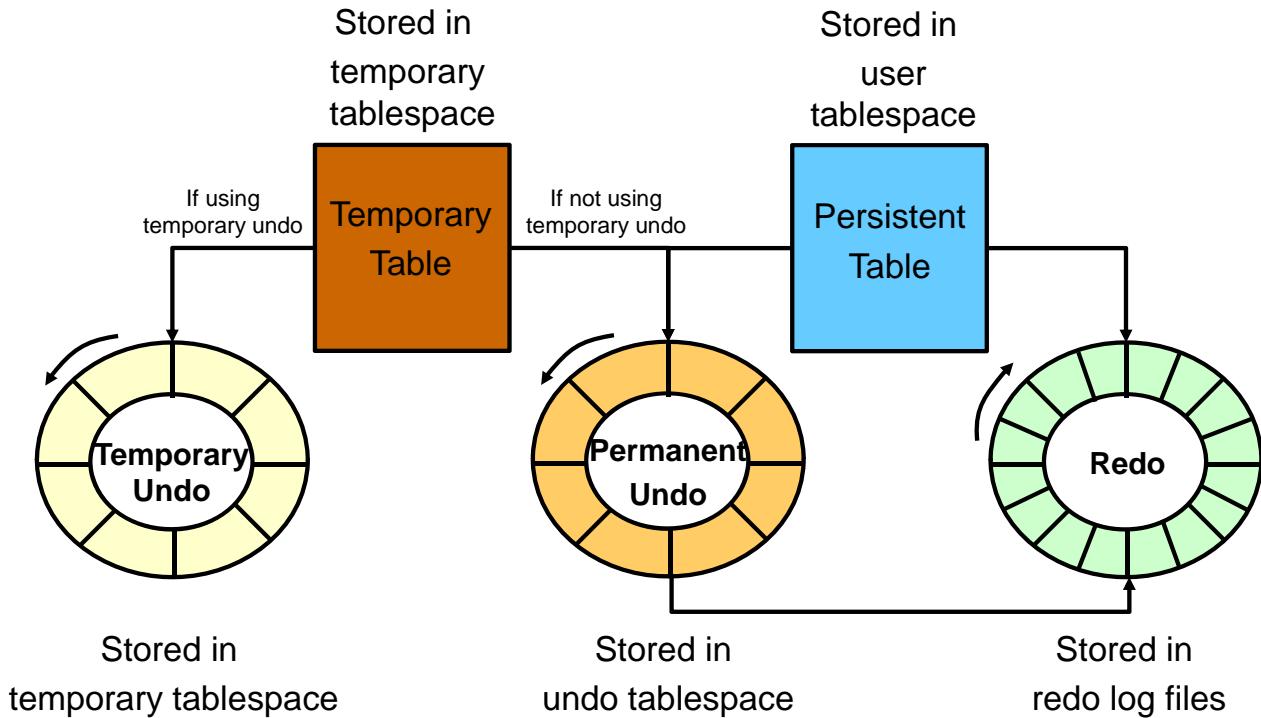
If you decide to change the undo tablespace to a fixed size, you must choose a large enough size to avoid the following two errors:

- DML failures (because there is not enough space to create the undo for new transactions)
- “Snapshot too old” errors (because there was insufficient undo data for read consistency)

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

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

Temporary Undo: Overview



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

Temporary Undo: Benefits

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Enabling temporary undo provides the following benefits:

- Temporary undo reduces the amount of undo stored in the undo tablespaces. Less undo in the undo tablespaces can result in more realistic undo retention period requirements for undo records.
- Temporary undo reduces the size of the redo log. Performance is improved because less data is written to the redo log, and components that parse redo log records, such as LogMiner, perform better because there is less redo data to parse.
- Temporary undo enables data manipulation language (DML) operations on temporary tables in a physical standby database with the Oracle Active Data Guard option. However, data definition language (DDL) operations that create temporary tables must be issued on the primary database.

Enabling Temporary Undo

- Enable temporary undo for a session:

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

- Enable temporary undo for the database instance:

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

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

Monitoring Temporary Undo

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

Viewing Undo Information

The screenshot shows the Oracle Enterprise Manager Database Express 12c interface. The title bar reads "ORACLE Enterprise Manager Database Express 12c". The top navigation bar includes links for Configuration, Storage, Security, and Performance. The main content area is titled "Undo Management Details". It contains several sections: "Configuration" (with "Undo Summary", "Tablespace", "Errors and Warnings", and "Advisor Findings" sections), "Undo Statistics Summary" (with "Analysis Period (Last Day)" and "Undo Retention Analysis" sections), and "Undo Statistics" (with various performance metrics). The "Errors and Warnings" section shows 0 errors and 0 warnings. The "Advisor Findings" section shows "Health" status with "No problems".

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can view undo information on the Undo Management Details page in Enterprise Manager Database Express.

Viewing Undo Activity



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

At the bottom of the Undo Management Details page, you can view additional statistical information:

- **Undo Generation Rate:** Displays the undo generation (in KB per second)
- **Undo Space Usage:** Shows the use of space in the tablespace
- **Steal Activity Breakdown:** Shows the number of attempts to steal expired undo blocks from other undo segments and attempts to obtain undo space by stealing unexpired extents from other transactions

Using the Undo Advisor

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can access the Undo Advisor in Enterprise Manager Cloud Control by clicking Performance > Advisors Home > Automatic Undo Management after selecting the database.

The middle part of the General tab contains the Undo Advisor analysis. It provides an estimate of the undo tablespace size required to satisfy a given undo retention.

Click Show Graph to see a graphical representation of the required tablespace size. You can click a point on the graph to see the tablespace size required to support the selected period.

To change the undo tablespace to a fixed size, click Edit Undo Tablespace, and then click Edit in the Datafile section.

Quiz

All you need to do to guarantee that all queries under 15 minutes will find the undo data needed for read consistency, is set the UNDO_RETENTION parameter to 15 minutes.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Viewing system activity
- Calculating undo tablespace sizing to support a 48-hour retention interval
- Modifying an undo tablespace to support a 48-hour retention interval



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

10

Managing Data Concurrency

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

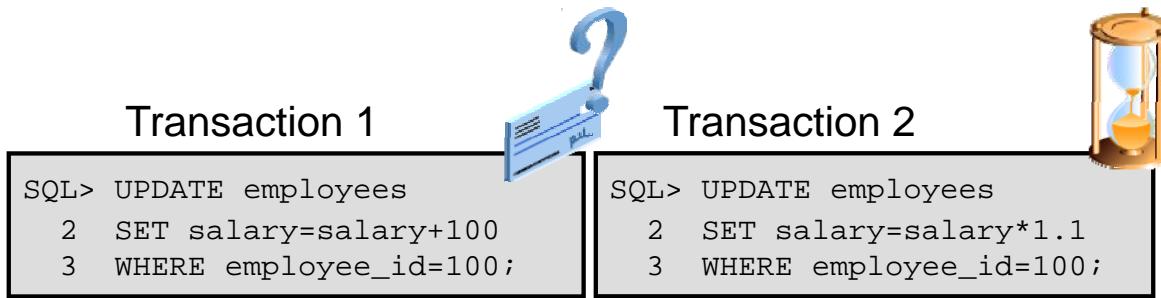
- Describe the locking mechanism and how Oracle manages data concurrency
- Monitor and resolve locking conflicts



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Locks

- Prevent multiple sessions from changing the same data at the same time
- Are automatically obtained at the lowest possible level for a given statement
- Do not escalate



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Before the database server allows a session to modify data, the session must first lock the data that is being modified. A lock gives the session exclusive control over the data so that no other transaction can modify the locked data until the lock is released.

Transactions can lock individual rows of data, multiple rows, or even entire tables. Oracle Database supports both manual and automatic locking. For automatically acquired locks, the lowest possible level of locking is chosen to minimize potential conflicts with other transactions.

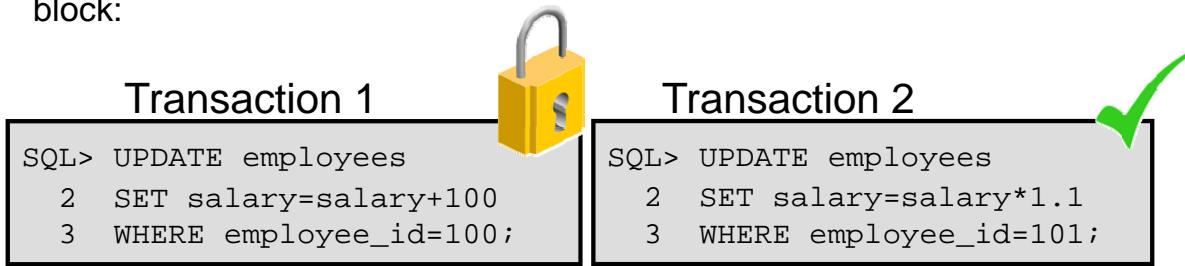
Note: There are many types of locks used by the Oracle server to maintain internal consistency. In this course, you will be concentrating only on locking that is used to protect rows and tables.

Locking Mechanism

- High level of data concurrency:
 - Row-level locks for inserts, updates, and deletes
 - No locks required for queries
- Automatic queue management
- Locks held until the transaction ends (with a commit or rollback operation)

Example

Assume that the rows for EMPLOYEE_ID 100 and 101 reside in the same block:



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The locking mechanism is designed to provide the maximum possible degree of data concurrency within the database. Transactions that modify data acquire row-level locks rather than block-level or table-level locks. Modifications to objects (such as table moves) obtain object-level locks rather than whole database or schema locks.

Data queries do not require a lock, and a query succeeds even if someone has locked the data (always showing the original, prelock value reconstructed from undo information).

When multiple transactions need to lock the same resource, the first transaction to request the lock obtains it. Other transactions wait until the first transaction completes. The queue mechanism is automatic and requires no administrator interaction.

All locks are released as transactions are completed (that is, when a COMMIT or ROLLBACK is issued). In the case of a failed transaction, the same background process that automatically rolls back any changes from the failed transaction releases all locks held by that transaction.

Data Concurrency

Time:	Transaction 1	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=100;
09:00:00	Transaction 2	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=101;
09:00:00	Transaction 3	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=102;
09:00:00
09:00:00	Transaction x	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=xxx;



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The lock mechanism defaults to a fine-grained, row-level locking mode. Different transactions can be updating different rows within the same table without interfering with one another.

Although the default model is to lock at the row level, Oracle Database supports manual locking at higher levels if needed:

```
SQL> LOCK TABLE employees IN EXCLUSIVE MODE;
Table(s) Locked.
```

With the preceding statement, any other transaction that tries to update a row in the locked table must wait until the transaction that issued the lock request completes. EXCLUSIVE is the strictest lock mode. The following are the other lock modes:

- **ROW SHARE:** Permits concurrent access to the locked table but prohibits sessions from locking the entire table for exclusive access
- **ROW EXCLUSIVE:** Is the same as ROW SHARE, but also prohibits locking in SHARE mode. The ROW EXCLUSIVE locks are automatically obtained when updating, inserting, or deleting data. ROW EXCLUSIVE locks allow multiple readers and one writer.
- **SHARE:** Permits concurrent queries but prohibits updates to the locked table. A SHARE lock is required (and automatically requested) to create an index on a table. However, online index creation requires a ROW SHARE lock that is used when building the index.

Share locks allow multiple readers and no writers. Share locks are also used transparently when deleting or updating rows in a parent table that has a child table with foreign key constraints on the parent.

- **SHARE ROW EXCLUSIVE:** Is used to query a whole table and to allow others to query rows in the table, but prohibits others from locking the table in SHARE mode or updating rows
- **EXCLUSIVE:** Permits queries on the locked table but prohibits any other activity on it. An EXCLUSIVE lock is required to drop a table.

Like any request for a lock, manual lock statements wait until all sessions that either already have locks or have previously requested locks release their locks. The `LOCK` command accepts a special argument that controls the waiting behavior, `NOWAIT`.

`NOWAIT` returns control to you immediately if the specified table is already locked by another session:

```
SQL> LOCK TABLE hr.employees IN SHARE MODE NOWAIT;
LOCK TABLE hr.employees IN SHARE MODE NOWAIT
*
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT specified
```

It is usually not necessary to manually lock objects. The automatic locking mechanism provides the data concurrency needed for most applications. Oracle recommends that you avoid using manual locks, especially when developing applications. Severe performance issues often occur from unnecessarily high locking levels.

DML Locks

Transaction 1

```
SQL> UPDATE employees  
  2  SET salary=salary*1.1  
  3  WHERE employee_id= 107;  
1 row updated.
```

Transaction 2

```
SQL> UPDATE employees  
  2  SET salary=salary*1.1  
  3  WHERE employee_id= 106;  
1 row updated.
```

Each DML transaction must acquire *two locks*:

- EXCLUSIVE row lock on the row or rows being updated
- Table lock (TM) in ROW EXCLUSIVE (RX) mode on the table containing the rows



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each DML transaction obtains two locks:

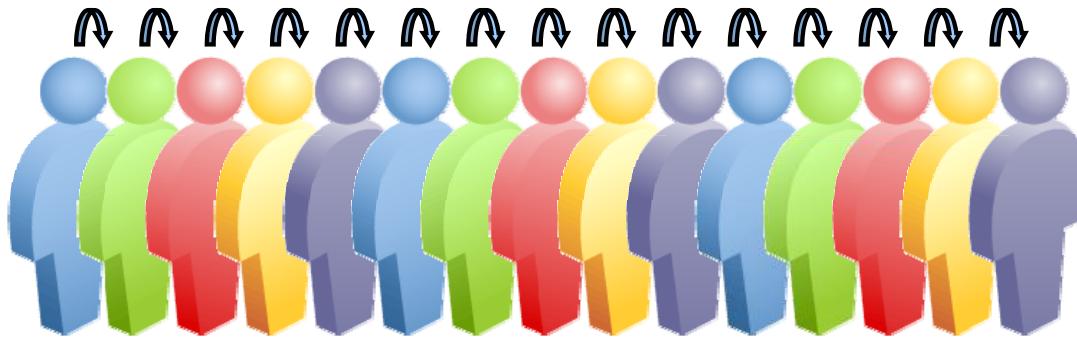
- An EXCLUSIVE row lock on the row or rows being updated
- A table lock (TM) in ROW EXCLUSIVE (RX) mode on the table being updated. This prevents another session from locking the whole table (possibly to drop or truncate it) while the change is being made. This mode is also called a subexclusive table lock (SX).

A ROW EXCLUSIVE lock on the table prevents DDL commands from changing the dictionary metadata in the middle of an uncommitted transaction. This preserves dictionary integrity and read consistency across the life of a transaction.

Enqueue Mechanism

The enqueue mechanism keeps track of:

- Sessions waiting for locks
- Requested lock mode
- Order in which sessions requested the lock



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Requests for locks are automatically queued. As soon as the transaction holding a lock is completed, the next session in line receives the lock.

The enqueue mechanism tracks the order in which locks are requested and the requested lock mode.

Sessions that already hold a lock can request that the lock be *converted* without having to go to the end of the queue. For example, suppose a session holds a `SHARE` lock on a table. The session can request that the `SHARE` lock be converted to an `EXCLUSIVE` lock. If no other transaction already has an `EXCLUSIVE` or `SHARE` lock on the table, the session holding the `SHARE` lock is granted an `EXCLUSIVE` lock without having to wait in the queue again.

Note: There are two categories of waiters for enqueues: those waiting without shared ownership and those with shared ownership that do not choose to escalate the lock level. Waiters in the second category are known as *converters*, which are always given priority over normal waiters even if they have been waiting less time.

Lock Conflicts

Transaction 1	Time	Transaction 2
UPDATE employees SET salary=salary+100 WHERE employee_id=100; 1 row updated.	9:00:00	UPDATE employees SET salary=salary+100 WHERE employee_id=101; 1 row updated.
UPDATE employees SET COMMISION_PCT=2 WHERE employee_id=101; Session waits enqueue due to lock conflict.	9:00:05 	SELECT sum(salary) FROM employees; SUM(SALARY) ----- 692634
Session still waiting!	16:30:00	Many selects, inserts, updates, and deletes during the last 7.5 hours, but no commits or rollbacks!
1 row updated. Session continues.	16:30:01	commit;



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Lock conflicts occur often but are usually resolved through time and the enqueue mechanism. In certain rare cases, a lock conflict may require administrator intervention. In the example in the slide, transaction 2 obtains a lock on a single row at 9:00:00 and neglects to commit, leaving the lock in place. Transaction 1 attempts to update the entire table, requiring a lock on all rows, at 9:00:05. Transaction 1 is blocked by transaction 2 until transaction 2 commits at 16:30:01.

A user attempting to perform transaction 1 would almost certainly contact the administrator for help in this case, and the DBA would have to detect and resolve the conflict.

Possible Causes of Lock Conflicts

- Uncommitted changes
- Long-running transactions
- Unnecessarily high locking levels



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The most common cause of lock conflicts is an uncommitted change, but there are a few other possible causes:

- **Long-running transactions:** Many applications use batch processing to perform bulk updates. These batch jobs are usually scheduled for times of low or no user activity, but in some cases, they may not have finished or may take too long to run during the low activity period. Lock conflicts are common when transaction and batch processing are being performed simultaneously.
- **Unnecessarily high locking levels:** Not all databases support row-level locking (Oracle added support for row-level locks in 1988 with release 6). Some databases still lock at the page or table level. Developers writing applications that are intended to run on many different databases often write their applications with artificially high locking levels so that Oracle Database behaves similarly to these less capable database systems. Developers who are new to Oracle also sometimes unnecessarily code in higher locking levels than are required by Oracle Database.

Detecting Lock Conflicts

- Select Blocking Sessions from the Performance menu.

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The top navigation bar includes links for Enterprise, Targets, Favorites, History, Setup, and ADMIN. A search bar for 'Search Target Name' is also present. The main content area is titled 'Blocking Sessions' and displays a hierarchical list of sessions. The first session listed is 'RPANDYA' with Session ID 37, which is blocking another session. The second session listed is 'DHAMBY' with Session ID 35. The table columns include Select, Username, Sessions Blocked, Session ID, Serial Number, SQL ID, Wait Class, Wait Event, P1 Value, P2 Value, P3 Value, and Seconds in Wait. The 'Wait Event' column for the RPANDYA session shows 'SQL*Net message from client' and for the DHAMBY session shows 'eng: TX - row lock contention'.

Select	Username	Sessions Blocked	Session ID	Serial Number	SQL ID	Wait Class	Wait Event	P1 Value	P2 Value	P3 Value	Seconds in Wait
<input type="radio"/>	Blocking Sessions										
<input checked="" type="radio"/>	RPANDYA	1	37	51299		Idle	SQL*Net message from client	1650815232	1	0	139
<input type="radio"/>	DHAMBY	0	35	23860	bk3sumaapsy1b	Application	eng: TX - row lock contention	1415053318	262169	1774	102

- Click the Session ID link to view information about the locking session.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Use the Blocking Sessions page in Enterprise Manager Cloud Control to locate lock conflicts. Conflicting lock requests are shown in a hierarchical layout, with the session holding the lock displayed at the top and, below that, any sessions that are queued for the lock.

For each session involved in the conflict, you are given the username, session ID, and number of seconds for which the session has been waiting. Drill down to the SQL ID to see the actual SQL statements that are currently being executed or requested by the session.

The Automatic Database Diagnostic Monitor (ADDM) also automatically detects lock conflicts and can advise you on inefficient locking trends.

Resolving Lock Conflicts

To resolve a lock conflict:

- Have the session holding the lock commit or roll back
- Terminate the session holding the lock (in an emergency)



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To resolve a lock conflict, the session holding the lock must release it. The best way to have the session release the lock is to contact the user and ask that the transaction be completed.

In an emergency, it is possible for the administrator to terminate the session holding the lock. Remember that when a session is killed, all work within the current transaction is lost (rolled back). A user whose session is killed must log in again and redo all work since the killed session's last commit.

Users whose sessions have been killed receive the following error the next time they try to issue a SQL statement:

ORA-03135: connection lost contact

Note: PMON can automatically kill sessions due to idle timeout. This can be implemented by using profiles or Resource Manager.

Resolving Lock Conflicts by Using SQL

SQL statements can be used to determine the blocking session and kill it.

1

```
SQL> SELECT sid, serial#, username  
2   FROM v$session WHERE sid IN  
3   (SELECT blocking_session FROM v$session);
```

Result:

SID	SERIAL#	USERNAME
144	8982	HR

2

```
SQL> ALTER SYSTEM KILL SESSION '144,8982' immediate;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Session manipulation can also be done by issuing SQL statements. The V\$SESSION view contains details of all connected sessions. The value in BLOCKING_SESSION is the session ID of the session that is blocking. If you query for SID and SERIAL# (where SID matches a blocking session ID), you have the information needed to perform the KILL SESSION operation.

Note: Database Resource Manager can be used to automatically log out sessions that block others and are idle.

Deadlocks

Transaction 1	Transaction 2
UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 1000;	9:00
UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 2000;	9:15
ORA-00060: Deadlock detected while waiting for resource	9:16



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A deadlock is a special example of a lock conflict. Deadlocks arise when two or more sessions wait for data that has been locked by the other. Because each is waiting for the other, neither can complete their transaction to resolve the conflict.

Oracle Database automatically detects deadlocks and terminates the statement with an error. The proper response to that error is either commit or roll back, which releases any other locks in that session so that the other session can continue its transaction.

In the example in the slide, transaction 1 must either commit or roll back in response to the deadlock detected error. If it commits, it must resubmit the second update to complete its transaction. If it performs a rollback, it must resubmit both statements to complete its transaction.

Quiz

The lock mechanism defaults to a fine-grained, row-level locking mode.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Quiz

When a deadlock occurs, Oracle database automatically:

- a. Waits 300 seconds before terminating both sessions
- b. Terminates one statement with an error in one session
- c. Terminates the statements with an error in both sessions
- d. Takes no action by default and leaves it to the DBA



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Describe the locking mechanism and how Oracle manages data concurrency
- Monitor and resolve locking conflicts



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Identifying locking conflicts
- Resolving locking conflicts



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

11

Implementing Oracle Database Auditing



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe DBA responsibilities for security and auditing
- Enable unified auditing
- Create unified audit policies
- Maintain the audit trail



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Database Security

A secure system ensures the confidentiality of the data that it contains. There are several aspects of security:

- Restricting access to data and services
- Authenticating users
- Monitoring for suspicious activity



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Database provides the industry's best framework for a secure system. But for that framework to be effective, the database administrator must follow best practices and continually monitor database activity.

Restricting Access to Data and Services

All users must not have access to all data. Depending on what is stored in your database, restricted access can be mandated by business requirements, by customer expectations, and (increasingly) by legal restrictions. Credit card information, health-care data, identity information, and so on must be protected from unauthorized access. The Oracle Database server provides extremely fine-grained authorization controls to limit database access. Restricting access must include applying the principle of least privilege.

Authenticating Users

To enforce access controls on sensitive data, the system must first know who is trying to access the data. Compromised authentication can render all other security precautions useless. The most basic form of user authentication is challenging users to provide something that they know, such as a password. Ensuring that passwords follow simple rules can greatly increase the security of your system. Stronger authentication methods include requiring users to provide something that they have, such as a token or public key infrastructure (PKI) certificate. An even stronger form of authentication is to identify users through a unique biometric characteristic such as a fingerprint, an iris scan, bone structure patterns, and so on. The Oracle Database server supports advanced authentication techniques (such as token-, biometric-, and certificate-based identification) through the Oracle Advanced Security option. User accounts that are not in use must be locked to prevent attempts to compromise authentication.

Monitoring for Suspicious Activity

Even authorized and authenticated users can sometimes compromise your system. Identifying unusual database activity (such as an employee who suddenly begins querying large amounts of credit card information, research results, or other sensitive information) can be the first step to detecting information theft. The Oracle Database server provides a rich set of auditing tools to track user activity and identify suspicious trends.

Monitoring for Compliance

- Monitoring or auditing must be an integral part of your security procedures.
- Review the following:
 - Mandatory auditing
 - Standard database auditing
 - Value-based auditing
 - Fine-grained auditing (FGA)



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Auditing, which means capturing and storing information about what is happening in the system, increases the amount of work the system must do. Auditing must be focused so that only events that are of interest are captured. Properly focused auditing has minimal impact on system performance. Improperly focused auditing can significantly affect performance.

- **Mandatory auditing:** All Oracle databases audit certain actions regardless of other audit options or parameters. The reason for mandatory audit logs is that the database needs to record some database activities, such as connections by privileged users.
- **Standard database auditing:** Select the objects and privileges that you want to audit and create the appropriate audit policies.
- **Value-based auditing:** Extends standard database auditing, capturing not only the audited event that occurred but also the actual values that were inserted, updated, or deleted. Value-based auditing is implemented through database triggers.
- **Fine-grained auditing (FGA):** Extends standard database auditing, capturing the actual SQL statement that was issued rather than only the fact that the event occurred.

Types of Activities to be Audited

You can audit the following types of activities:

- User accounts, roles, and privileges
- Object actions
- Application context values
- Oracle Data Pump
- Oracle Database Real Application Security
- Oracle Database Vault
- Oracle Label Security
- Oracle Recovery Manager
- Oracle SQL*Loader direct path events



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Through the use of auditing policies, you can configure audit settings for the following activities:

- Logging on to the database and the use of privileges and roles
- Executing SQL statements against specific database objects
- Application context values
- Utilities and features:
 - Oracle Data Pump
 - Oracle Database Real Application Security
 - Oracle Database Vault
 - Oracle Label Security
 - Oracle Recovery Manager
 - Oracle SQL*Loader Direct Load

Mandatorily Audited Activities

The following activities are audited:

- CREATE/ALTER/DROP AUDIT POLICY
- AUDIT/NOAUDIT
- EXECUTE of:
 - DBMS_FGA
 - DBMS_AUDIT_MGMT
- ALTER TABLE against AUDSYS audit trail table
- Top-level statements by administrative users (SYS, SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM) until the database opens



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following audit-related activities are mandatorily audited:

- CREATE/ALTER/DROP AUDIT POLICY
- AUDIT/NOAUDIT
- EXECUTE of the DBMS_FGA PL/SQL package
- EXECUTE of the DBMS_AUDIT_MGMT PL/SQL package
- ALTER TABLE attempts on the AUDSYS audit trail table
- Top-level statements by SYS, SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM until the database opens. After the database opens these users are audited based on the defined audit settings.

Understanding Auditing Implementation

- *Mixed mode auditing* is the default when a new Oracle Database 12c database is created.
- Mixed mode auditing enables the use of:
 - Pre-Oracle Database 12c auditing features
 - *Unified auditing* features of Oracle Database 12c
- The recommendation from Oracle is to migrate to unified auditing.
- Query V\$OPTION to determine if the database has been migrated to unified auditing:

```
SELECT value FROM v$option  
WHERE parameter = 'Unified Auditing'
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Prior to Oracle Database 12c, audit records from various sources were stored in different locations. Oracle Database 12c supports *unified auditing*, in which all audit records are stored in a single audit table.

When you create a new Oracle Database 12c database, mixed mode auditing is enabled. This mode enables you to use the auditing features available before Oracle Database 12c and also the unified auditing features. Mixed mode auditing is enabled by default through the ORA_SECURECONFIG predefined auditing policy for newly created databases.

If you are upgrading a database to Oracle Database 12c, you must manually migrate to unified auditing to use the unified auditing features.

Oracle Corporation recommends that you migrate to unified auditing.

Administering the Roles Required for Auditing

A user must be granted one of the following roles to perform auditing:

- **AUDIT_ADMIN** enables the user to:
 - Create unified and fine-grained audit policies
 - Execute the AUDIT and NOAUDIT SQL statements
 - View audit data
 - Manage the audit trail (table in the AUDSYS schema)
- **AUDIT_VIEWER** enables the user to:
 - View and analyze audit data

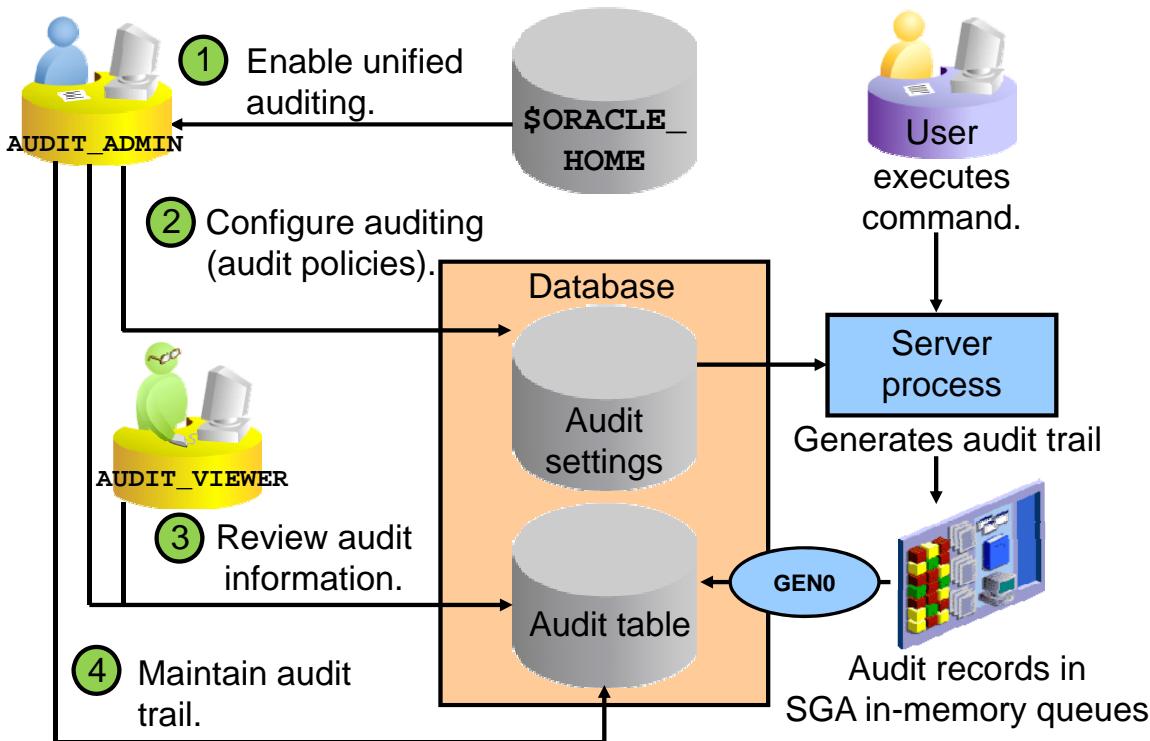


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Users must be granted the appropriate privilege to configure auditing and view audit data. To support separation of duty, two default roles are provided:

- **AUDIT_ADMIN**: Enables the grantee to configure auditing settings, create and administer audit policies (unified and find-grained), and view and analyze audit data. This role is typically granted to a security administrator.
- **AUDIT_VIEWER**: Enables the grantee to view and analyze audit data. This role is typically granted to external auditors.

Database Auditing: Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To use the unified audit trail features, you must first enable unified auditing.

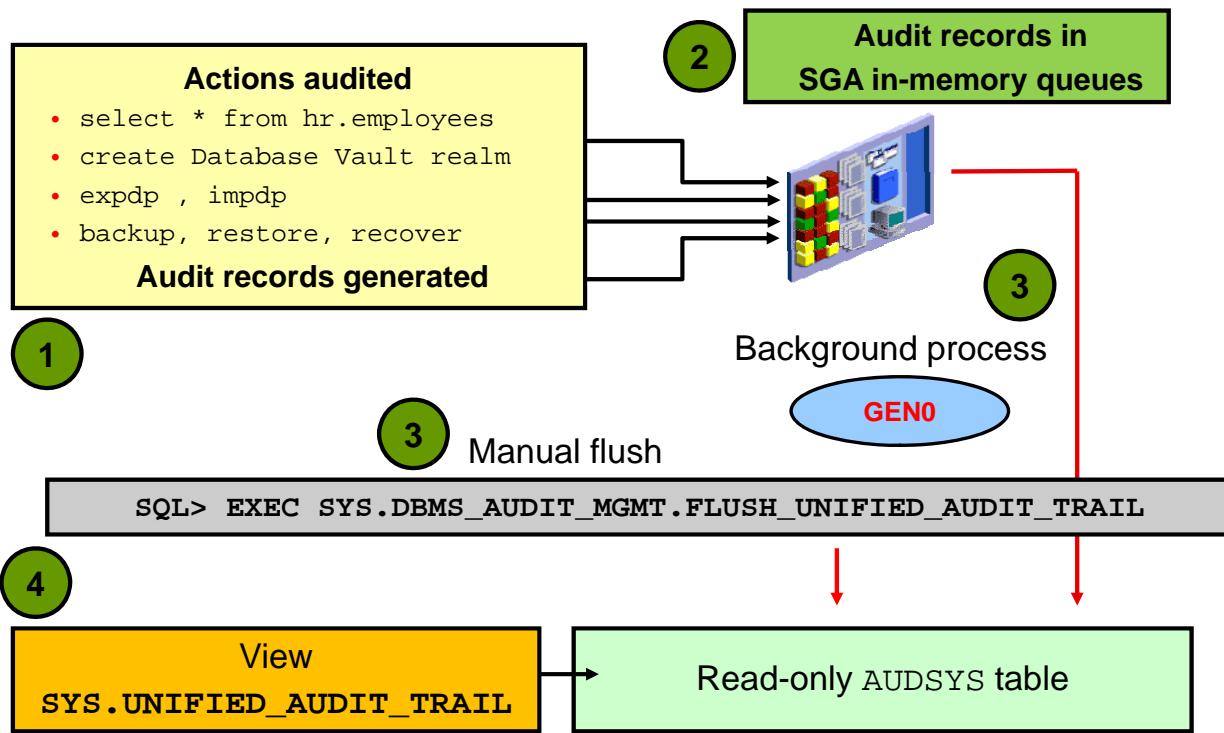
You must then create and enable unified audit policies to specify which activities should be audited.

When a user executes a command or performs an activity that is defined in an auditing policy, audit records are generated. The audit records are written to an audit table in the AUDSYS schema and can be viewed by querying the UNIFIED_AUDIT_TRAIL view.

Maintaining the audit trail is an important administrative task. Depending on the focus of the audit options, the audit trail can grow very large very quickly. If not properly maintained, the audit trail can create so many records that it affects the performance of the system. Audit overhead is directly related to the number of records that are produced.

Detailed information on all these steps is provided later in this lesson.

Understanding the Audit Architecture



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The default mode of operation for unified auditing is *queued write mode*. Audit records are written to a queue in the System Global Area (SGA). The named queues in 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(s).

Each client has two SGA queues so that a client can continue to write to the second queue while the first queue is being written to the table.

You can use the following procedure to explicitly flush the queues to disk so that you view the audit trail records immediately:

```
SQL> EXEC SYS.DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL
```

Enabling Unified Auditing

1. In SQL*Plus, shut down the database instance:

```
SQL> SHUTDOWN IMMEDIATE
```

2. Shut down the listener:

```
$ lsnrctl stop
```

3. At the operating system prompt, enable the unified auditing executable:

```
$ cd $ORACLE_HOME/rdbms/lib  
$ make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME
```

4. Restart the listener:

```
$ lsnrctl start
```

5. In SQL*Plus, restart the database instance:

```
SQL> STARTUP
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Before enabling the unified auditing executable, shut down the database instance and the listener.

Change to the \$ORACLE_HOME/rdbms/lib directory and execute the following command:

```
make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME
```

The make command is used to relink the Oracle executable with a different set of libraries to enable unified auditing.

After the make command completes, restart the listener and the database instance.

You can log in to SQL*Plus and verify that unified auditing has been enabled as follows:

```
SQL> select value  
  2  from v$option  
  3  where parameter = 'Unified Auditing';
```

VALUE

TRUE

Configuring Auditing

Method	Description
Unified audit policies	Group audit settings into a policy
Default unified audit policies	Three default policies: ORA_SECURECONFIG ORA_DATABASE_PARAMETER_AUDIT ORA_ACCOUNT_MGMT_AUDIT
Fine-grained audit policies	Define specific conditions that must be met for auditing to take place



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can configure auditing by grouping audit settings into a unified audit policy.

Oracle Database includes three predefined unified audit policies, which include commonly used security audit settings:

- ORA_SECURECONFIG: Contains the same default audit settings as Oracle Database 11g. This policy is enabled by default for unified auditing and mixed-mode auditing environments. Refer to the *Oracle Database Security Guide 12c* for a complete list of settings.
- ORA_DATABASE_PARAMETER_AUDIT: Contains audit settings for commonly used Oracle Database commands that set parameters (ALTER DATABASE, ALTER SYSTEM, and CREATE SPFILE)
- ORA_ACCOUNT_MGMT_AUDIT: Contains audit settings for commonly used user account and privilege commands (CREATE USER, ALTER USER, DROP USER, CREATE ROLE, DROP ROLE, ALTER ROLE, SET ROLE, GRANT, and REVOKE)

Fine-grained audit policies enable you to define specific conditions that generate an audit record.

Additional information about these methods is provided later in the lesson.

Using Enterprise Manager Cloud Control

Audit Policy	Comments	System Privileges	Roles	Component Actions	Object Actions	Conditional	Enabled	Audit	Users
ORA_ACCOUNT_MGMT		0	0	9	0				
ORA_GIS_RECOMMENDATIONS	Audit policy containing aud...	7	0	19	0				
ORA_DATABASE_PARAMETER		0	0	3	0				
ORA_DV_AUDPOL		0	0	0	1768				
ORA_LOGON_FAILURES		0	0	1	0		<input checked="" type="checkbox"/>	By	ALL USERS
ORA_RAS_POLICY_MGMT		0	0	33	0				
ORA_RAS_SESSION_MGMT		0	0	14	0				
ORA_SECURECONFIG		29	0	16	1		<input checked="" type="checkbox"/>	By	ALL USERS

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

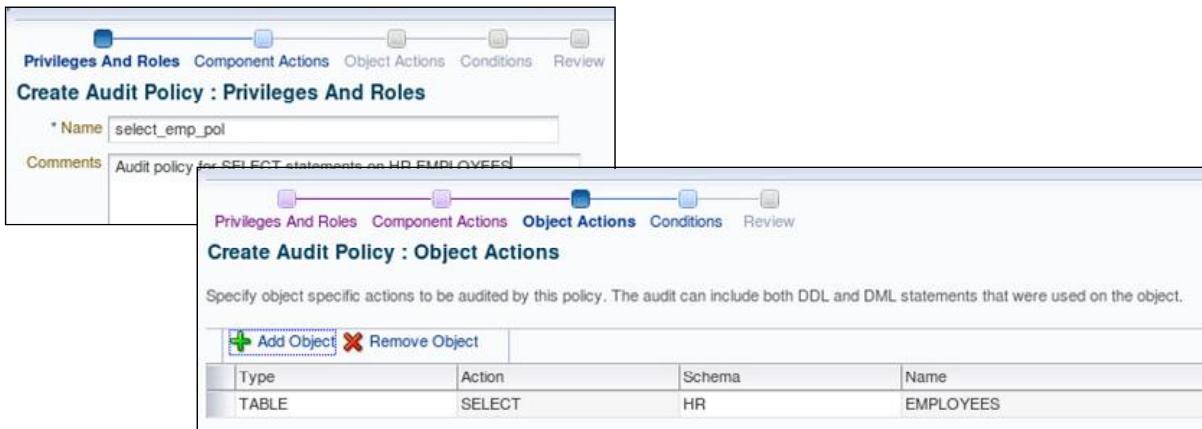
You can use Enterprise Manager Cloud Control to view and manage audit settings. Access the Audit Settings page by expanding the Security menu on your database home page and selecting Audit Settings.

Creating a Unified Audit Policy

- Use the CREATE AUDIT POLICY statement:

```
CREATE AUDIT POLICY select_emp_pol  
  ACTIONS select on hr.employees
```

- Use Enterprise Manager Cloud Control:



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Use the CREATE AUDIT POLICY statement to create a unified audit policy. The example in the slide shows how to create an audit policy named SELECT_EMP_POL. This policy specifies that SELECT statements against the HR.EMPLOYEES table will be audited.

You can also create audit policies by using Enterprise Manager Cloud Control. On the Audit Settings page, click Create to invoke the wizard.

Creating an Audit Policy: System-Wide Audit Options

- System privileges:

```
CREATE AUDIT POLICY audit_syspriv_pol1  
PRIVILEGES SELECT ANY TABLE, CREATE LIBRARY
```

- Actions:

```
CREATE AUDIT POLICY audit_actions_pol2  
ACTIONS AUDIT, ALTER TRIGGER
```

- Roles:

```
CREATE AUDIT POLICY audit_role_pol3  
ROLES mgr_role
```

- System privileges, actions, and roles:

```
CREATE AUDIT POLICY audit_mixed_pol4  
PRIVILEGES DROP ANY TABLE  
ACTIONS      CREATE TABLE, DROP TABLE, TRUNCATE TABLE  
ROLES        emp_role
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can create an audit policy with system-wide or object-specific audit options.

The system-wide options can be of three types:

- The privilege audit option audits all events that exercise the specified system privilege, as in the first example in 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 MGR_ROLE role, as in the third example.

You can configure privilege, action, and role audit options together in the same audit policy, as shown in the fourth example.

You can find a list of auditable system-wide options in the SYS.AUDITABLE_SYSTEM_ACTIONS table.

Creating an Audit Policy: Object-Specific Actions

Create audit policies based on object-specific options.

```
CREATE AUDIT POLICY audit_objpriv_pol5  
  ACTIONS SELECT, UPDATE, LOCK ON hr.employees
```

```
CREATE AUDIT POLICY audit_objpriv_pol6  
  ACTIONS ALL
```

```
CREATE AUDIT POLICY audit_objpriv_pol7  
  ACTIONS EXECUTE, GRANT ON hr.raise_salary_proc
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Object-specific options are the second type of audit options. 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 in 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 an Audit Policy: Specifying Conditions

- Condition and evaluation **PER SESSION**

```
CREATE AUDIT POLICY audit_mixed_pol5
ACTIONS RENAME ON hr.employees,ALTER ON hr.jobs,
WHEN 'SYS_CONTEXT (''USERENV'', ''SESSION_USER'')='JIM''
EVALUATE PER SESSION
```

- Condition and evaluation **PER STATEMENT**

```
CREATE AUDIT POLICY audit_objpriv_pol6
ACTIONS ALTER ON OE.ORDERS
WHEN 'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'')='OE''
EVALUATE PER STATEMENT
```

- Condition and evaluation **PER INSTANCE**

```
CREATE AUDIT POLICY audit_objpriv_pol7
ROLES dba
WHEN SYS_CONTEXT(''USERENV'', ''INSTANCE_NAME'')='sales''
EVALUATE PER INSTANCE
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Audit policies can evaluate the condition per statement, once per session or once per instance.

The first example in 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 “sales.” 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.

Enabling and Disabling Audit Policies

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 © 2014, Oracle and/or its affiliates. All rights reserved.

After creating the audit policy, you must enable it 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.

To disable an audit policy, use the NOAUDIT command.

Altering a Unified Audit Policy

- Use the ALTER AUDIT POLICY statement:

```
ALTER AUDIT POLICY select_emp_pol  
ADD ACTIONS select on hr.job_history
```

- Use Enterprise Manager Cloud Control:

Audit Policy	System Privileges	Roles	Component Actions	Object Actions	Conditional	Enabled	Audit	Users
ORA_SECURECONFIG	29	0	15	0		<input checked="" type="checkbox"/>	All	
ORA_RAS_POLICY_MGMT	0	0	32	0				
ORA_RAS_SESSION_MGMT	0	0	14	0				
ORA_ACCOUNT_MGMT	0	0	9	0				
ORA_DATABASE_PARAMETER	0	0	3	0				
SELECT_EMP_POL	0	0	0	1		<input checked="" type="checkbox"/>	By	HR

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can change most properties in a unified audit policy, except a CONTAINER setting. You can make changes to enabled and disabled audit policies. You do not need to re-enable an audit policy after altering it.

For an object unified audit policy, the new audit settings take place immediately after it has been altered, for both the active and subsequent user sessions. If you alter system audit options, or audit conditions of the policy, then they are activated for new user sessions, but not the current user session.

To alter an audit policy by using Enterprise Manager Cloud Control, select the policy on the Audit Settings page and click Edit.

Viewing Audit Policy Information

```
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

```
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

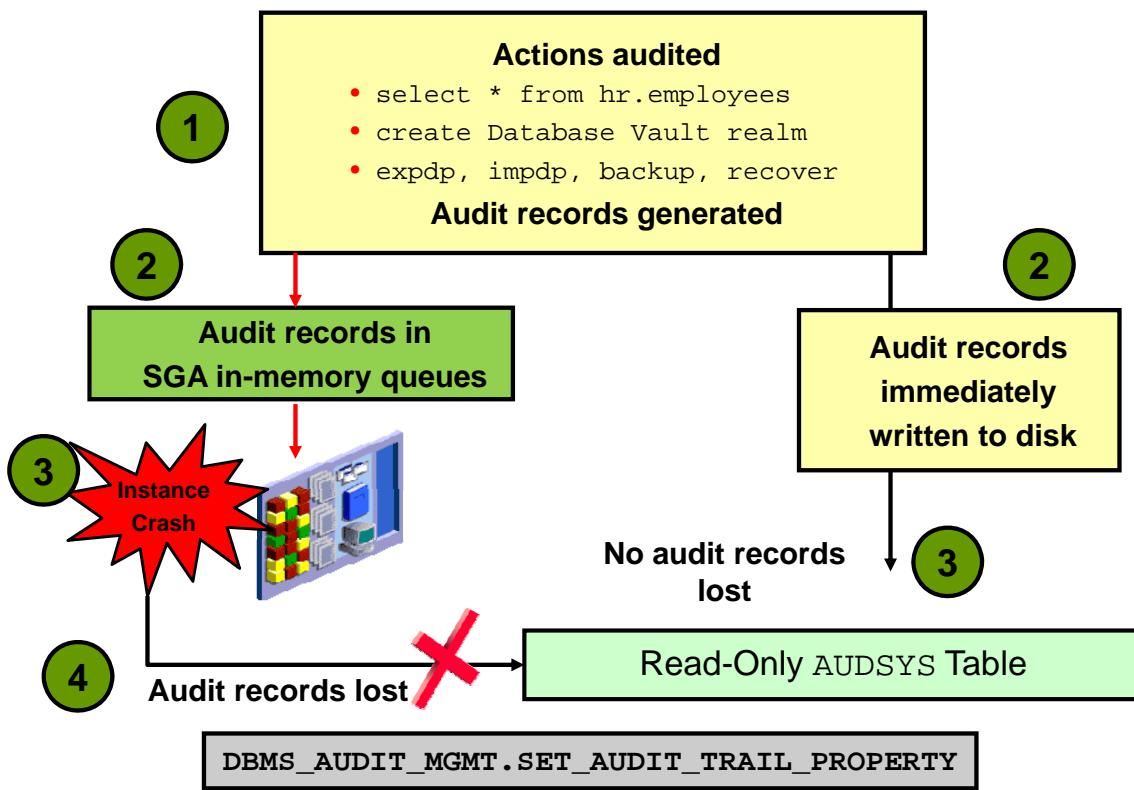


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Query AUDIT_UNIFIED_POLICIES to view a list of the existing audit policies. The CONDITION_EVAL_OPT column defines when the condition, if any, is evaluated.

Query AUDIT_UNIFIED_ENABLED_POLICIES to view a list of enabled audit policies. The ENABLED_OPT column indicates whether 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 indicate whether the policy generates audit records only when the user's actions succeed or fail.

Setting the Write Mode for Audit Trail Records



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

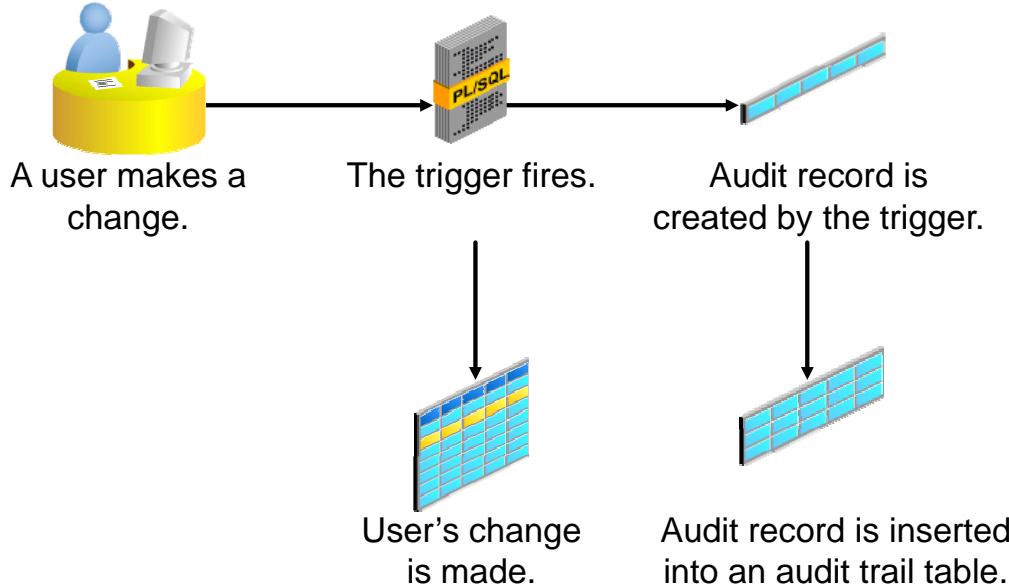
The in-memory queuing infrastructure means that the audit records are not written immediately to the AUDSYS table, but are queued in the SGA. In cases such as an instance crash, *queued* audit records can be lost. Hence, the audit trail must be configurable to indicate the tolerance level. The following two modes are based on different levels of loss tolerance:

- Immediate-Write mode: The audit records are written immediately.

```
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 written to disk. This is the default mode.

```
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);
```

Value-Based Auditing



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Value-based auditing leverages database triggers (event-driven PL/SQL constructs) to capture changed values in objects.

When a user inserts, updates, or deletes data from a table with the appropriate trigger attached, the trigger works in the background to copy audit information to a table that is designed to contain the audit information. Value-based auditing tends to degrade performance more than standard database auditing because the audit trigger code must be executed each time the insert, update, or delete operation occurs. The degree of degradation depends on the efficiency of the trigger code. Value-based auditing must be used only in situations in which the information captured by standard database auditing is insufficient.

Value-based auditing is implemented by user or third-party code. The Oracle database provides the PL/SQL constructs to allow value-based audit systems to be built.

The key to value-based auditing is the audit trigger, which is simply a PL/SQL trigger that is constructed to capture audit information.

Example of a typical audit trigger:

```
CREATE OR REPLACE TRIGGER system.hrsalary_audit
  AFTER UPDATE OF salary
  ON hr.employees
  REFERENCING NEW AS NEW OLD AS OLD
  FOR EACH ROW
BEGIN
  IF :old.salary != :new.salary THEN
    INSERT INTO system.audit_employees
    VALUES (sys_context('userenv','os_user'), sysdate,
            sys_context('userenv','ip_address'),
            :new.employee_id ||
            ' salary changed from '||:old.salary|||
            ' to '||:new.salary);
  END IF;
END;
/
```

This trigger focuses auditing to capture changes to the salary column of the `hr.employees` table. When a row is updated, the trigger checks the salary column. If the old salary is not equal to the new salary, the trigger inserts an audit record into the `audit_employees` table (created via a separate operation in the `SYSTEM` schema). The audit record includes the username, the IP address from which the change is made, the primary key identifying which record is changed, and the actual salary values that are changed.

Database triggers can also be used to capture information about user connections in cases where standard database auditing does not gather sufficient data. With login triggers, the administrator can capture data that identifies the user who is connecting to the database. Examples include the following:

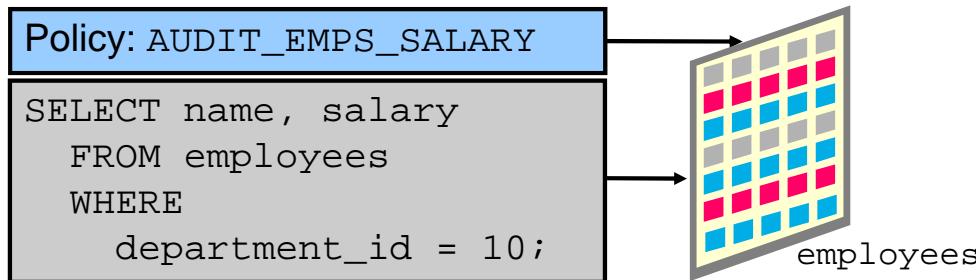
- IP address of the person logging in
- First 48 characters of the program name that is used to connect to the instance
- Terminal name that is used to connect to the instance

For a complete list of user parameters, see the section titled “`SYS_CONTEXT`” in the *Oracle Database SQL Reference*.

Value-based triggers have been superseded in many cases by the fine-grained auditing (FGA) feature.

Fine-Grained Auditing

- Monitors data access on the basis of content
- Audits SELECT, INSERT, UPDATE, DELETE, and MERGE
- Can be linked to one or more columns in a table or view
- May execute a procedure
- Is administered with the DBMS_FGA package



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Database auditing records the fact that an operation has occurred but does not capture information about the statement that caused the operation. Fine-grained auditing (FGA) extends that capability to enable the capture of actual SQL statements that query or manipulate data.

FGA also allows auditing to be more narrowly focused than standard or value-based database auditing.

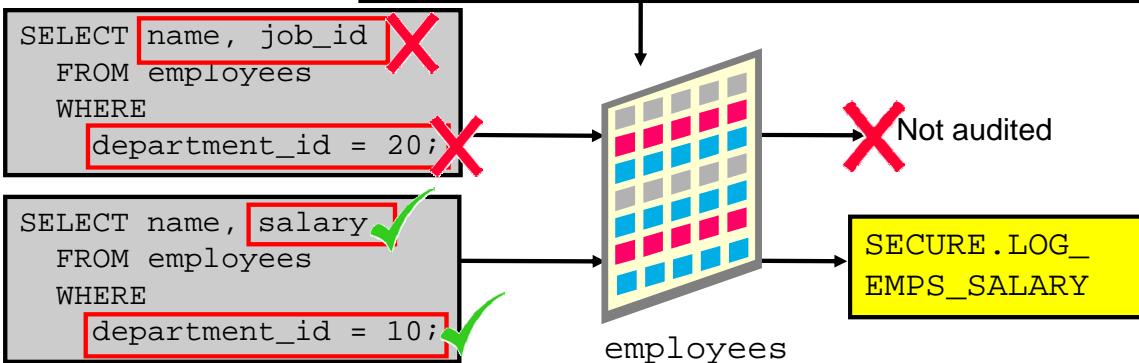
FGA options can be focused by individual columns in a table or view, and can even be conditional so that audits are captured only if certain administrator-defined specifications are met. More than one relevant column is supported for an FGA policy. By default, if any one of these columns is present in the SQL statement, it is audited. DBMS_FGA.ALL_COLUMNS and DBMS_FGA.ANY_COLUMNS are provided to audit on the basis of whether any or all of the relevant columns are used in the statement.

Use the DBMS_FGA PL/SQL package to create an audit policy on the target table or view. If any of the rows returned from a query block match the audited column and the specified audit condition, an audit event causes an audit record to be created and stored in the audit trail. As an option, the audit event can also execute a procedure. FGA automatically focuses auditing at the statement level. A SELECT statement that returns thousands of rows thus generates only one audit record.

FGA Policy

- Defines:
 - Audit criteria
 - Audit action
- Is created with DBMS_FGA .ADD_POLICY

```
dbms_fga.add_policy (
  object_schema  => 'HR',
  object_name    => 'EMPLOYEES',
  policy_name   => 'audit_emps_salary',
  audit_condition=> 'department_id=10',
  audit_column    => 'SALARY,COMMISSION_PCT',
  handler_schema  => 'secure',
  handler_module  => 'log_emps_salary',
  enable          => TRUE,
  statement_types => 'SELECT,UPDATE' );
```



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows the creation of a fine-grained auditing policy with the DBMS_FGA.ADD_POLICY procedure, which accepts the following arguments.

Policy Name

You assign each FGA policy a name when you create it. The example in the slide names the policy AUDIT_EMPS_SALARY by using the following argument:

```
policy_name => 'audit_emps_salary'
```

Audit Condition

The audit condition is a SQL predicate that defines when the audit event must fire. In the slide example, all rows in department 10 are audited by using the following condition argument:

```
audit_condition => 'department_id = 10'
```

Note: Fine-grained auditing looks at the result set of the query, so with the FGA policy shown in the slide, queries that return rows matching the policy specifications will cause an audit record to be created. For example, in the query "select * from employees", all rows including those having "10" in DEPARTMENT_ID may be returned, so an audit row is created.

Audit Column

The audit column defines the data that is being audited. An audit event occurs if this column is included in the `SELECT` statement or if the audit condition allows the selection. The example in the slide audits two columns by using the following argument:

```
audit_column => 'SALARY,COMMISION_PCT'
```

This argument is optional. If it is not specified, only the `AUDIT_CONDITION` argument determines whether an audit event must occur.

Object

The object is the table or view that is being audited. It is passed as two arguments:

- The schema that contains the object
- The name of the object

The example in the slide audits the `hr.employees` table by using the following arguments:

```
object_schema => 'hr'  
object_name => 'employees'
```

Handler

An optional event handler is a PL/SQL procedure that defines additional actions that must be taken during auditing. For example, the event handler can send an alert page to the administrator. If it is not defined, an audit event entry is inserted into the audit trail. If an audit event handler is defined, the audit entry is inserted into the audit trail and the audit event handler is executed.

The audit event entry includes the FGA policy that caused the event, the user executing the SQL statement, and the SQL statement and its bind variables.

The event handler is passed as two arguments:

- The schema that contains the PL/SQL program unit
- The name of the PL/SQL program unit

The example in the slide executes the `SECURE.LOG_EMPS_SALARY` procedure by using the following arguments:

```
handler_schema => 'secure'  
handler_module => 'log_emps_salary'
```

By default, the audit trail always writes the SQL text and SQL bind information to LOBs. The default can be changed (for example, if the system would suffer performance degradation).

Status

The status indicates whether the FGA policy is enabled. In the slide example, the following argument enables the policy:

```
enable => TRUE
```

Audited DML Statement: Considerations

- Records are audited if the FGA predicate is satisfied and the relevant columns are referenced.
- DELETE statements are audited regardless of columns specified.
- MERGE statements are audited with the underlying INSERT, UPDATE, and DELETE generated statements.

```
UPDATE hr.employees  
SET salary = 1000  
WHERE commission_pct = .2;
```

Not audited because none
of the employees are in
department 10



```
UPDATE hr.employees  
SET salary = 1000  
WHERE employee_id = 200;
```

Audited because the
employee is in department
10



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

With an FGA policy defined for DML statements, a DML statement is audited if the data rows (both new and old) that are being manipulated meet the policy predicate criteria.

However, if relevant columns are also specified in the policy definition, the statement is audited when the data meets the FGA policy predicate and the statement references the relevant columns defined.

For DELETE statements, specifying relevant columns during policy definition is not useful because all columns in a table are touched by a DELETE statement. Therefore, a DELETE statement is always audited regardless of the relevant columns.

MERGE statements are supported by FGA. The underlying INSERT, UPDATE, and DELETE statements are audited if they meet the defined INSERT, UPDATE, or DELETE FGA policies.

Using the previously defined FGA policy, the first statement is not audited whereas the second one is. None of the employees in department 10 receive a commission, but employee_id=200 specifies an employee in department 10.

FGA Guidelines

- To audit all rows, use a `null` audit condition.
- To audit all columns, use a `null` audit column.
- Policy names must be unique.
- The audited table or view must already exist when you create the policy.
- If the audit condition syntax is invalid, an `ORA-28112` error is raised when the audited object is accessed.
- If the audited column does not exist in the table, no rows are audited.
- If the event handler does not exist, no error is returned and the audit record is still created.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For the `SELECT` statements, FGA captures the statement itself and not the actual rows. However, when FGA is combined with Flashback Query, the rows can be reconstructed as they existed at that point in time.

For more details about the `DBMS_FGA` package, see the *Oracle Database PL/SQL Packages and Types Reference*.

Archiving and Purging the Audit Trail

- Periodically archive and purge the audit trail to prevent it from growing too large.
- Create an archive by:
 - Copying audit trail records to a database table
 - Using Oracle Audit Vault
- Purge the audit trail by:
 - Creating and scheduling a purge job to run at a specified time by using the DBMS_AUDIT_MGMT.CREATE_PURGE_JOB PL/SQL procedure
 - Manually by using the DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL PL/SQL procedure



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Basic maintenance of the audit trail includes reviewing the audit records and removing older records. The audit trail can grow to fill the available storage if not reviewed periodically and archived and purged. If the database audit trail fills the tablespace, audited actions do not complete.

Purging Audit Trail Records

- 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 © 2014, 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 includes the CONTAINER attribute. 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.

USE_LAST_ARCH_TIMESTAMP specifies whether the last archived time stamp should be used for determining 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 because 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

Top-level statements performed before the database opens by administrative users such as SYS and SYSDBA are mandatorily audited.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe DBA responsibilities for security and auditing
- Enable unified auditing
- Create unified audit policies
- Maintain the audit trail



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice: Overview

This practice covers the following topics:

- Enabling unified auditing
- Granting the AUDIT_ADMIN role
- Creating and enabling a unified audit policy
- Testing the audit policy
- Reviewing audit information
- Maintaining the audit trail



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.