

Oracle Database 11g: Administration Workshop II

Volume II • Student Guide

D50079GC20

Edition 2.0

September 2010

D62544

ORACLE®

Author

Maria Billings

**Technical Contributors
and Reviewers**

Christian Bauwens

Yanti Chang

Timothy Chien

Joe Fong

Andy Fortunak

Gerlinde Frenzen

Mark Fuller

Peter Fusek

Joel Goodman

Vimala Jacob

Dominique Jeunot

Pete Jones

Fukue Kawabe

Donna Keesling

Sean Kim

Achiel Langers

Gwen Lazenby

Jerry Lee

Deidre Matishak

Bill Millar

Lakshmi Naraparreddi

Ira Singer

Ranbir Singh

James Spiller

Matt Taylor

Branislav Valny

Jean-Francois Verrier

Editors

Nita Pavitrans

Raj Kumar

Graphic Designer

Satish Bettegowda

Publisher

Jayanthi Keshavamurthy

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

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

Trademark Notice

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

Contents

I Introduction

- Course Objectives 1-2
- Suggested Schedule 1-3
- Oracle Database 11g: “g” Stands for Grid 1-4
- Grid Infrastructure for Single-Instance 1-6
- Course Examples: HR Sample Schema 1-8

1 Core Concepts and Tools of the Oracle Database

- Objectives 1-2
- Naming the Core Components of an Oracle Database Server 1-3
- Oracle Database Server Architecture Overview 1-4
- Instance-Database Configurations 1-6
- Naming the Memory Structures of an Oracle Database Instance 1-7
- Oracle Database Memory Structures 1-8
- Process Architecture 1-10
- Process Structures 1-11
- Adding Process Names 1-13
- Process Startup Sequence 1-14
- Database Storage Architecture 1-15
- Logical and Physical Database Structures 1-17
- Automatic Storage Management 1-19
- ASM Storage Components 1-20
- ASM Instance 1-21
- DBA Configuration Tools 1-23
- Management Framework and Related DBA Tools 1-25
- Facilitating Database Management with Oracle Restart 1-26
- Notes: Facilitating Database Management with Oracle Restart 1-27
- Quiz 1-28
- Summary 1-29

2 Configuring for Recoverability

- Objectives 2-2
- Purpose of Backup and Recovery Functionality 2-3
- Typical Backup and Recovery Tasks 2-4
- Oracle Backup and Recovery Solutions 2-5

Oracle Backup Solutions	2-6
Terminology Review	2-7
What You Already Know: Oracle-Suggested Backup	2-9
Using Recovery Manager	2-10
Types of RMAN Commands	2-11
Job Commands: Example	2-12
Configuring Your Database for Backup and Recovery Operations	2-13
ARCHIVELOG Mode	2-14
Configuring ARCHIVELOG Mode	2-15
Configuring Archive Log Destinations	2-16
Guaranteeing Archive Log Success	2-17
Specifying a Retention Policy	2-19
A Recovery Window Retention Policy: Example	2-21
Using a Fast Recovery Area	2-22
Defining a Fast Recovery Area	2-24
Fast Recovery Area Space Management	2-25
Fast Recovery Area Space Usage	2-27
What Is Done Automatically for You	2-29
Monitoring the FRA	2-30
Benefits of Using a Fast Recovery Area	2-31
Quiz	2-32
Summary	2-34
Practice 2 Overview: Configuring for Recoverability	2-35

3 Using the RMAN Recovery Catalog

Objectives	3-2
RMAN Repository Data Storage: Comparison of Options	3-3
Storing Information in the Recovery Catalog	3-4
Reasons to Use a Recovery Catalog	3-5
Creating the Recovery Catalog: Three Steps	3-6
Configuring the Recovery Catalog Database	3-7
Creating the Recovery Catalog Owner	3-8
Creating the Recovery Catalog	3-9
Managing Target Database Records in the Recovery Catalog	3-10
Registering a Database in the Recovery Catalog	3-11
Using Enterprise Manager to Register a Database	3-12
Unregistering a Target Database from the Recovery Catalog	3-13
Cataloging Additional Backup Files	3-14
Recovery Catalog Resynchronization: Concepts	3-16
Manually Resynchronizing the Recovery Catalog	3-17
Using RMAN Stored Scripts	3-18

- Executing RMAN Stored Scripts 3-19
- Maintaining RMAN Stored Scripts 3-20
- Backing Up the Recovery Catalog 3-21
- Re-Creating an Unrecoverable Recovery Catalog 3-22
- Exporting and Importing the Recovery Catalog 3-23
- Upgrading and Dropping the Recovery Catalog 3-24
- IMPORT CATALOG Command 3-25
- Creating and Using Virtual Private Catalogs 3-27
- Using RMAN Virtual Private Catalogs 3-28
- Recovery Catalogs Summary 3-30
- Quiz 3-32
- Summary 3-34
- Practice 3 Overview: Using the RMAN Recovery Catalog 3-35

4 Configuring Backup Settings

- Objectives 4-2
- Configuring Persistent Settings for RMAN 4-3
- Viewing Persistent Settings 4-4
- Control File Autobackups 4-5
- Managing Persistent Settings 4-7
- Using a Media Manager 4-8
- Specifying a Backup Destination 4-10
- Configuring and Allocating Channels 4-11
- Creating Duplexed Backup Sets 4-12
- Creating Duplexed Backup Sets Using CONFIGURE BACKUP COPIES 4-13
- Backup Optimization 4-14
- Saving Backup Space with Unused Block Compression 4-16
- Compressing Backups 4-17
- Using RMAN Backup Compression 4-18
- Encrypting Backups 4-19
- Quiz 4-20
- Summary 4-22
- Practice 4 Overview: Configuring Backup Specifications 4-23

5 Creating Backups with RMAN

- Objectives 5-2
- Creating Backup Sets 5-3
- Creating Image Copies 5-4
- Creating a Whole Database Backup 5-6
- RMAN Backup Types 5-8
- Fast Incremental Backup 5-10

- Enabling Fast Incremental Backup 5-11
- Monitoring Block Change Tracking 5-12
- Performing Proxy Copies 5-13
- Creating Duplexed Backup Sets Using `BACKUP COPIES` 5-14
- Creating Backups of Backup Sets 5-15
- Backing Up Read-Only Tablespaces 5-16
- Configuring Backup and Restore for Very Large Files 5-17
- Creating RMAN Multisection Backups 5-18
- Archival Backups: Concepts 5-19
- Creating Archival Backups with EM 5-21
- Creating Archival Backups with RMAN 5-22
- Managing Archival Database Backups 5-23
- Backing Up Recovery Files 5-24
- Managing Backups: Reporting 5-25
- Managing Backups: Dynamic Performance Views 5-27
- Using Enterprise Manager to View Backup Reports 5-28
- Managing Backups: Cross-Checking and Deleting 5-29
- Quiz 5-30
- Summary 5-32
- Practice 5 Overview: Creating Backups 5-33

6 Restore and Recovery Tasks

- Objectives 6-2
- Restoring and Recovering 6-3
- Causes of File Loss 6-4
- Critical Versus Noncritical 6-5
- Automatic Tempfile Recovery 6-6
- Log Group Status: Review 6-7
- Recovering from the Loss of a Redo Log Group 6-8
- Clearing a Log File 6-9
- Recovering from a Lost Index Tablespace 6-10
- Re-Creating Indexes 6-11
- Authentication Methods for Database Administrators 6-13
- Re-creating a Password Authentication File 6-14
- Comparing Complete and Incomplete Recovery 6-16
- Complete Recovery Process 6-17
- Point-in-Time Recovery Process 6-18
- Recovering a Read-Only Tablespace 6-20
- Recovering `NOLOGGING` Database Objects 6-21
- Recovering from the Loss of All Control File Copies: Overview 6-22
- Recovering the Control File to the Default Location 6-23

Quiz 6-24

Summary 6-26

7 Using RMAN to Perform Recovery

Objectives 7-2

Using RMAN RESTORE and RECOVER Commands 7-3

Performing Complete Recovery: Loss of a Noncritical Data File in ARCHIVELOG Mode 7-4

Performing Complete Recovery: Loss of a System-Critical Data File in ARCHIVELOG Mode 7-5

Recovering Image Copies 7-6

Recovering Image Copies: Example 7-7

Performing a Fast Switch to Image Copies 7-8

Using SET NEWNAME for Switching Files 7-9

Substitution Variables for SET NEWNAME 7-10

Performing Restore and Recovery of a Database in NOARCHIVELOG Mode 7-11

Using Restore Points 7-12

Performing Point-in-Time Recovery 7-13

Performing Recovery with a Backup Control File 7-15

Recovery from Loss of Server Parameter File 7-16

Restoring the Server Parameter File from the Control File Autobackup 7-17

Restoring the Control File from Autobackup 7-18

Using Incremental Backups to Recover a Database in NOARCHIVELOG Mode 7-20

Restoring and Recovering the Database on a New Host 7-21

Preparing to Restore the Database to a New Host 7-22

Restoring the Database to a New Host 7-23

Performing Disaster Recovery 7-27

Quiz 7-29

Summary 7-31

Practice 7 Overview: Using RMAN to Perform Recovery 7-32

8 Monitoring and Tuning RMAN

Objectives 8-2

Parallelization of Backup Sets 8-3

Monitoring RMAN Sessions 8-5

Monitoring RMAN Job Progress 8-7

Interpreting RMAN Message Output 8-9

Using the DEBUG Option 8-10

Interpreting RMAN Error Stacks 8-11

Tuning RMAN 8-12

- RMAN Multiplexing 8-14
- Allocating Disk Buffers: Example 8-15
- Allocating Tape Buffers 8-16
- Comparing Synchronous and Asynchronous I/O 8-18
- Monitoring RMAN Job Performance 8-20
- Asynchronous I/O Bottlenecks 8-21
- Synchronous I/O Bottlenecks 8-22
- Channel Tuning 8-23
- Tuning the BACKUP Command 8-25
- Tuning RMAN Backup Performance 8-27
- Setting LARGE_POOL_SIZE 8-28
- Tuning RMAN Tape Streaming Performance Bottlenecks 8-29
- Quiz 8-31
- Summary 8-33
- Practice 8 Overview: Monitoring and Tuning RMAN 8-34

9 Diagnosing the Database

- Objectives 9-2
- Data Recovery Advisor 9-3
- Data Failures 9-6
- Data Failure: Examples 9-7
- Data Recovery Advisor RMAN Command-Line Interface 9-8
- Listing Data Failures 9-9
- Advising on Repair 9-11
- Executing Repairs 9-12
- Classifying (and Closing) Failures 9-13
- Data Recovery Advisor Views 9-14
- Best Practice: Proactive Checks 9-15
- What Is Block Corruption? 9-16
- Block Corruption Symptoms: ORA-01578 9-17
- How to Handle Corruption 9-18
- Setting Parameters to Detect Corruption 9-19
- Block Media Recovery 9-21
- Prerequisites for Block Media Recovery 9-22
- The RECOVER . . . BLOCK Command 9-23
- Automatic Diagnostic Workflow 9-24
- Automatic Diagnostic Repository 9-25
- The ADR Command-Line Tool (ADRCI) 9-26
- The V\$DIAG_INFO View 9-27
- Location for Diagnostic Traces 9-28

Health Monitor: Overview 9-29
Running Health Checks Manually: PL/SQL Example 9-30
Viewing HM Reports Using the ADRCI Utility 9-31
Quiz 9-32
Summary 9-36
Practice 9 Overview: Diagnosing the Database 9-37

10 Using Flashback Technology I

Objectives 10-2
Flashback Technology 10-3
Transactions and Undo 10-4
Guaranteeing Undo Retention 10-5
Preparing Your Database for Flashback 10-6
Using Flashback Technology to Query Data 10-8
Flashback Query 10-9
Flashback Query: Example 10-10
Flashback Version Query 10-11
Flashback Version Query: Considerations 10-12
Quiz 10-13
Flashback Table: Overview 10-15
Flashback Table 10-16
Enabling Row Movement on a Table 10-17
Performing Flashback Table 10-18
Flashback Table: Considerations 10-19
Quiz 10-20
Flashback Transaction Query 10-21
Using Enterprise Manager to Perform Flashback Transaction Query 10-22
Flashback Transaction Query: Considerations 10-23
Flashback Transaction 10-24
Prerequisites 10-25
Flashing Back a Transaction 10-26
Possible Workflow 10-27
Flashback Transaction Wizard 10-28
Choosing Other Back-out Options 10-29
Final Steps Without EM 10-31
Quiz 10-32
Summary 10-33
Practice 10 Overview: Performing Flashback Transaction Backout 10-34

11 Using Flashback Technology II

- Objectives 11-2
- Oracle Total Recall Overview 11-3
- Setup Process 11-4
- How Total Recall Works 11-5
- Oracle Total Recall Scenario 11-6
- Transparent Schema Evolution 11-8
- Full Schema Evolution 11-9
- Restrictions 11-10
- Guidelines 11-11
- Viewing Flashback Data Archives 11-12
- Quiz 11-13
- Flashback Drop and the Recycle Bin 11-15
- Recycle Bin 11-16
- Restoring Tables from the Recycle Bin 11-18
- Recycle Bin: Automatic Space Reclamation 11-19
- Recycle Bin: Manual Space Reclamation 11-20
- Bypassing the Recycle Bin 11-21
- Querying the Recycle Bin 11-22
- Quiz 11-23
- Summary 11-24
- Practice 11 Overview: Using Flashback Technology 11-25

12 Performing Flashback Database

- Objectives 12-2
- Flashback Database 12-3
- Flashback Database Architecture 12-4
- Configuring Flashback Database 12-5
- What You Need to Do 12-6
- Flashback Database: Examples 12-7
- Flashback Database Considerations 12-8
- Monitoring Flashback Database 12-9
- Monitoring Flashback Database with EM 12-11
- Guaranteed Restore Points 12-12
- Flashback Database and Guaranteed Restore Points 12-13
- Quiz 12-15
- Summary 12-17
- Practice 12 Overview: Working with Flashback Database 12-18

13 Managing Memory

- Objectives 13-2
- Memory Management: Overview 13-3
- Reviewing Oracle Database Memory Structures 13-4
- Buffer Cache 13-6
- Using Multiple Buffer Pools 13-8
- Shared Pool 13-10
- Large Pool 13-11
- Java Pool and Streams Pool 13-12
- Redo Log Buffer 13-13
- Automatic Memory Management: Overview 13-14
- Oracle Database Memory Parameters 13-15
- Monitoring Automatic Memory Management 13-16
- Efficient Memory Usage: Guidelines 13-18
- Memory Tuning Guidelines for the Library Cache 13-20
- Automatic Shared Memory Management: Overview 13-22
- How ASMM Works 13-23
- Enabling Automatic Shared Memory Management 13-24
- Disabling ASMM 13-25
- Program Global Area (PGA) 13-26
- Using the `V$PARAMETER` View 13-28
- Quiz 13-29
- Summary 13-30
- Practice 13 Overview: Using AMM to Correct a Memory Allocation Problem 13-31

14 Managing Database Performance

- Objectives 14-2
- Tuning Activities 14-3
- Performance Planning 14-4
- Instance Tuning 14-6
- Performance Tuning Methodology 14-7
- Performance Monitoring 14-8
- Performance Tuning Data 14-9
- Optimizer Statistics Collection 14-10
- Statistic Preferences: Overview 14-12
- Using Statistic Preferences 14-13
- Setting Global Preferences with Enterprise Manager 14-14
- Oracle Wait Events 14-15
- Instance Statistics 14-16
- Monitoring Session Performance 14-18
- Displaying Session-Related Statistics 14-19

- Displaying Service-Related Statistics 14-20
- Troubleshooting and Tuning Views 14-21
- Dictionary Views 14-22
- Automatic Workload Repository 14-23
- Using Automatic Workload Repository Views 14-25
- Real Application Testing Overview: Database Replay 14-26
- The Big Picture 14-27
- Quiz 14-28
- Summary 14-29
- Practice 14 Overview: Monitoring Instance Performance 14-30

15 Managing Performance by SQL Tuning

- Objectives 15-2
- SQL Tuning 15-3
- SQL Advisors 15-4
- Automatic SQL Tuning Results 15-5
- Implement Automatic Tuning Recommendations 15-6
- SQL Tuning Advisor: Overview 15-7
- Using the SQL Tuning Advisor 15-8
- SQL Tuning Advisor Options 15-9
- SQL Tuning Advisor Recommendations 15-10
- Using the SQL Tuning Advisor: Example 15-11
- Duplicate SQL 15-12
- SQL Access Advisor: Overview 15-13
- Typical SQL Access Advisor Session 15-14
- Workload Source 15-15
- Recommendation Options 15-16
- Reviewing Recommendations 15-18
- SQL Performance Analyzer: Overview 15-19
- SQL Performance Analyzer: Use Cases 15-20
- Using SQL Performance Analyzer 15-21
- Quiz 15-22
- Summary 15-26
- Practice 15 Overview: Managing Performance by SQL Tuning 15-27

16 Managing Resources

- Objectives 16-2
- Database Resource Manager: Overview 16-3
- Database Resource Manager: Concepts 16-4
- Why Use Resource Manager 16-5
- Default Maintenance Resource Manager Plan 16-7

Example: <code>DEFAULT_PLAN</code>	16-8
Potential Work Flow	16-9
Specifying Resource Plan Directives	16-11
Resource Allocation Methods for Resource Plans	16-12
Comparison of <code>EMPHASIS</code> and <code>RATIO</code>	16-13
Active Session Pool Mechanism	16-15
Setting the Active Session Pool	16-16
Specifying Thresholds	16-18
Setting Idle Timeouts	16-19
Limiting CPU Utilization at the Database Level	16-20
Limiting CPU Utilization at the Server Level: Instance Caging	16-22
Instance Caging Examples	16-23
Monitoring Instance Caging	16-24
Resource Consumer Group Mapping	16-25
Activating a Resource Plan	16-27
Database Resource Manager Information	16-28
Monitoring the Resource Manager	16-29
Quiz	16-32
Summary	16-33
Practice 16 Overview: Using the Resource Manager	16-34

17 Automating Tasks with the Scheduler

Objectives	17-2
Simplifying Management Tasks	17-3
Core Components	17-4
Your Basic Work Flow	17-5
Quiz	17-7
Persistent Lightweight Jobs	17-8
Using a Time-Based or Event-Based Schedule	17-9
Creating a Time-Based Job	17-10
Creating an Event-Based Schedule	17-12
Creating Event-Based Schedules with Enterprise Manager	17-13
Creating an Event-Based Job	17-14
Event-Based Scheduling	17-15
Creating Complex Schedules	17-17
Quiz	17-18
Using Email Notification	17-19
Adding and Removing Email Notifications	17-20
Creating Job Chains	17-21
Example of a Chain	17-23
Advanced Scheduler Concepts	17-24

- Job Classes 17-25
- Windows 17-27
- Prioritizing Jobs Within a Window 17-28
- Creating a Job Array 17-29
- Quiz 17-31
- Creating a File Watcher and an Event-Based Job 17-32
- Enabling File Arrival Events from Remote Systems 17-34
- Scheduling Remote Database Jobs 17-35
- Creating Remote Database Jobs 17-36
- Scheduling Multiple Destination Jobs 17-37
- Viewing Scheduler Meta Data 17-38
- Quiz 17-40
- Summary 17-41
- Practice 17 Overview: Automating Tasks with the Scheduler 17-42

18 Managing Space

- Objectives 18-2
- Space Management: Overview 18-3
- Block Space Management 18-4
- Row Chaining and Migration 18-5
- Quiz 18-7
- Free Space Management Within Segments 18-8
- Types of Segments 18-9
- Allocating Extents 18-10
- Allocating Space 18-11
- Creating Tables Without Segments 18-12
- Controlling Deferred Segment Creation 18-13
- Restrictions and Exceptions 18-14
- Additional Automatic Functionality 18-15
- Quiz 18-16
- Table Compression: Overview 18-17
- Compression for Direct-Path Insert Operations 18-18
- OLTP Compression for DML Operations 18-20
- Specifying Table Compression 18-21
- Using the Compression Advisor 18-22
- Using the `DBMS_COMPRESSION` Package 18-23
- Compressing Table Data 18-24
- Proactive Tablespace Monitoring 18-25
- Thresholds and Resolving Space Problems 18-26
- Monitoring Tablespace Space Usage 18-27
- Shrinking Segments 18-28

- Results of Shrink Operation 18-29
- Reclaiming Space Within ASSM Segments 18-30
- Segment Advisor: Overview 18-31
- Segment Advisor 18-32
- Implementing Recommendations 18-33
- Automatic Segment Advisor 18-34
- Manual Segment Shrink Using EM 18-35
- Shrinking Segments Using SQL 18-36
- Managing Resumable Space Allocation 18-37
- Using Resumable Space Allocation 18-38
- Resuming Suspended Statements 18-40
- What Operations Are Resumable? 18-42
- Quiz 18-43
- Summary 18-44
- Practice 18 Overview: Managing Storage 18-45

19 Managing Space for the Database

- Objectives 19-2
- Database Storage 19-3
- Supporting 4-KB Sector Disks 19-4
- Using 4-KB Sector Disks 19-5
- Specifying the Disk Sector Size 19-6
- Quiz 19-7
- Transporting Tablespaces 19-10
- Concept: Minimum Compatibility Level 19-11
- Minimum Compatibility Level 19-12
- Transportable Tablespace Procedure 19-13
- Determining the Endian Format of a Platform 19-14
- Using the RMAN `CONVERT` Command 19-16
- Transportable Tablespaces with Enterprise Manager 19-17
- Transporting Databases 19-20
- Database Transportation Procedure: Source System Conversion 19-21
- Database Transportation Procedure: Target System Conversion 19-22
- Database Transportation: Considerations 19-23
- Quiz 19-24
- Summary 19-25
- Practice 19 Overview: Managing Space for the Database 19-26

20 Duplicating a Database

- Objectives 20-2
- Using a Duplicate Database 20-3
- Choosing Database Duplication Techniques 20-4
- Duplicating an Active Database 20-5
- Duplicating a Database with a Target Connection 20-6
- Duplicating a Database with Recovery Catalog Without Target Connection 20-7
- Duplicating a Database Without Recovery Catalog or Target Connection 20-8
- Creating a Backup-Based Duplicate Database 20-9
- Creating an Initialization Parameter File for the Auxiliary Instance 20-10
- Specifying New Names for Your Destination 20-11
- Using the `SET NEWNAME` Clauses 20-12
- Substitution Variables for `SET NEWNAME` 20-13
- Specifying Parameters for File Naming 20-14
- Starting the Instance in `NOMOUNT` Mode 20-16
- Ensuring That Backups and Archived Redo Log Files Are Available 20-17
- Allocating Auxiliary Channels 20-18
- Understanding the RMAN Duplication Operation 20-19
- Specifying Options for the `DUPLICATE` Command 20-21
- Using Additional `DUPLICATE` Command Options 20-22
- Using EM to Clone a Database 20-23
- Quiz 20-24
- Summary 20-25
- Practice 20 Overview: Duplicating a Database 20-26

Appendix A: Practices and Solutions

Appendix B: Performing Tablespace Point-in-Time Recovery

- Objectives B-2
- Tablespace Point-in-Time Recovery (TSPITR): Concepts B-3
- Tablespace Point-in-Time Recovery (TSPITR): Terminology B-4
- Tablespace Point-in-Time Recovery: Architecture B-5
- When to Use TSPITR B-7
- Preparing for TSPITR B-8
- Determining the Correct Target Time B-9
- Determining the Tablespaces for the Recovery Set B-10
- Identifying Relationships That Span Recovery Set Boundaries B-11
- Identifying Objects That Will Be Lost B-12
- Performing Basic RMAN TSPITR B-13
- Performing Fully Automated TSPITR B-14

Using Image Copies for Faster TSPITR Performance	B-15
Using Enterprise Manager to Perform TSPITR	B-16
RMAN TSPITR Processing	B-17
Performing RMAN TSPITR with an RMAN-Managed Auxiliary Instance	B-19
Performing RMAN TSPITR Using Your Own Auxiliary Instance	B-20
Troubleshooting RMAN TSPITR	B-21
Summary	B-22

Appendix C: Performing User-Managed Backup and Recovery

Objectives	C-2
Types of Backup and Recovery Practices	C-3
Performing a User-Managed Backup of the Database	C-4
The Need for Backup Mode	C-5
Identifying Files to Manually Backup	C-6
Manually Backing Up a NOARCHIVELOG Database	C-7
Manually Backing Up an ARCHIVELOG Database	C-8
Backing Up the Control File	C-9
Performing User-Managed Complete Database Recovery: Overview	C-10
Performing Complete Closed Database Recovery: Overview	C-11
Identifying Recovery-Related Files	C-12
Restoring Recovery-Related Files	C-13
Applying Redo Data	C-15
Performing Complete Open Database Recovery	C-16
Performing User-Managed Incomplete Recovery: Overview	C-18
Choosing an Incomplete Recovery Method	C-19
Performing User-Managed Incomplete Recovery	C-20
Performing User-Managed Incomplete Recovery: Steps	C-22
User-Managed Time-Based Recovery: Example	C-23
User-Managed Cancel-Based Recovery: Example	C-25
Summary	C-27

Appendix D: Managing the ASM Instance

Objectives	D-2
ASM Benefits for Administrators	D-3
ASM Instance	D-4
ASM Components: ASM Instance—Primary Processes	D-6
ASM Instance Initialization Parameters	D-7
Interaction Between Database Instances and ASM	D-9
ASM Instance: Dynamic Performance Views	D-10
ASM System Privileges	D-11
Using Enterprise Manager to Manage ASM Users	D-12

Starting and Stopping ASM Instances Using SQL*Plus	D-13
Starting and Stopping ASM Instances Using <code>srvctl</code>	D-15
Starting and Stopping ASM Instances Using <code>asmcmd</code>	D-16
Disk Group Overview	D-17
ASM Disks	D-18
Allocation Units	D-19
ASM Files	D-20
Extent Maps	D-21
Striping Granularity	D-22
Fine Grained Striping	D-23
ASM Failure Groups	D-25
Stripe and Mirror Example	D-26
Failure Example	D-27
Managing Disk Groups	D-28
Creating and Dropping Disk Groups Using SQL*Plus	D-29
Adding Disks to Disk Groups	D-30
Miscellaneous <code>ALTER</code> Commands	D-31
ASM Management Using Enterprise Manager	D-32
ASM Disk Group Compatibility	D-33
ASM Disk Group Attributes	D-35
Using Enterprise Manager to Edit Disk Group Attributes	D-36
Retrieving ASM Metadata	D-37
ASM Fast Mirror Resync Overview	D-38
Summary	D-39

11

Using Flashback Technology II

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

> **Total Recall**
Flashback Drop

After completing this lesson, you should be able to:

- Describe and use Oracle Total Recall
 - Creating and enabling a Flashback Data Archive (FDA)
 - Managing FDAs
 - Viewing metadata
- Describe and use flashback recycle bins
 - Restore dropped tables from the recycle bin
 - Manage space usage in the recycle bin
 - Query the recycle bin

ORACLE

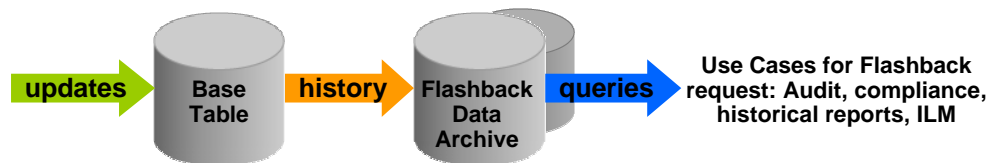
Copyright © 2009, Oracle. All rights reserved.

Oracle Total Recall Overview

Automated tracking of historical database changes:

- Enable at the table level with your specified retention period.
- All subsequent changes are transparently stored and tamper proof.
- Records older than retention period are automatically removed.
- Use Flashback technologies to retrieve history.

```
SELECT ... AS OF TIMESTAMP...  
SELECT ... VERSIONS BETWEEN TIMESTAMP and TIMESTAMP...
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Total Recall Overview

The Oracle Total Recall option in Oracle Database 11g (also known as Flashback Data Archive) provides a mechanism for tracking changes to production databases that is secure, efficient, easy to use, and application transparent.

With Oracle Total Recall technology, you can automatically track and store the data in tables enabled for Flashback Data Archive. This ensures that flashback queries obtain SQL-level access to the versions of database objects without getting a snapshot-too-old error.

A Flashback Data Archive provides the ability to track and store all transactional changes to a “tracked” table over its life time. It is no longer necessary to build this intelligence into your application. You can use Oracle Total Recall for compliance, audit reports, data analysis, and decision-support systems. The Flashback Data Archive background process starts with the database.

Use case examples:

- Audit support: Find duplicate insurance claims from the last year.
- Compliance support: Monitor stock trading during a quiet period.
- Information Lifecycle Management (ILM): Guarantee immutable access to patient history.
- Retention policy enforcement: Automatically purge records older than five years.
- Historical reporting: Retrieve a client’s credit and payment history.
- Error Recovery: Restore records erroneously removed or updated.

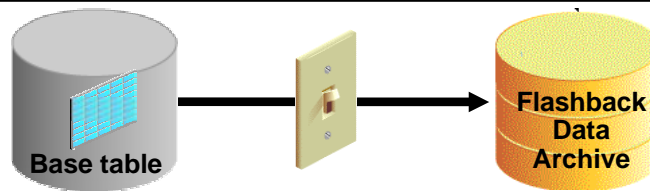
Setup Process

1. Create a new tablespace to hold the FDA.
2. With the FLASHBACK ARCHIVE ADMINISTER system privilege: Create a Flashback Data Archive, assign it to the tablespace, and specify its retention period.

```
CREATE FLASHBACK ARCHIVE fdal  
TABLESPACE fda_tbs1 QUOTA 10M RETENTION 1 YEAR;
```

3. With the FLASHBACK ARCHIVE object privilege: Alter the base tables to enable archiving and assign it to a flashback archive.

```
ALTER TABLE HR.EMPLOYEES FLASHBACK ARCHIVE fdal;
```



ORACLE

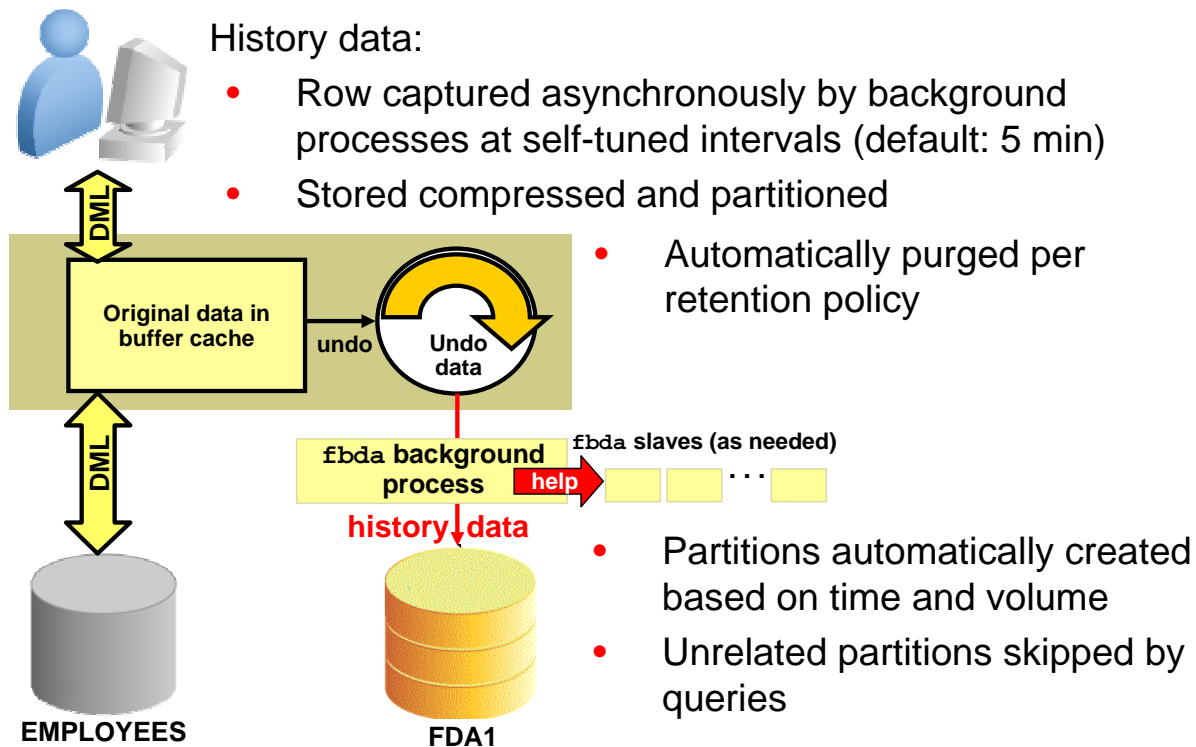
Copyright © 2009, Oracle. All rights reserved.

Flashback Data Archive Process

A Flashback Data Archive consists of one or more tablespaces. You can have multiple Flashback Data Archives. They are configured with retention duration. Based on your retention duration requirements, you should create different Flashback Data Archives—for example, one for all records that must be kept for two years, another for all records that must be kept for five years. The database will automatically purge all historical information on the day after the retention period expires.

1. Create a tablespace for your Flashback Data Archive. The size depends on the base table and the expected DML and DDL activity.
2. Create a Flashback Data Archive with retention time. Data archived in the Flashback Data Archive is retained for the retention time. This task requires the FLASHBACK ARCHIVE ADMINISTER system privilege. If different retention periods are needed, different archives must be created.
3. Enable flashback archiving (and then disable it again) for a (whole) table. This task requires the FLASHBACK ARCHIVE object privilege. Although flashback archiving is enabled for a table, some DDL statements are not allowed on that table. By default, flashback archiving is off for any table.

How Total Recall Works



ORACLE

Copyright © 2009, Oracle. All rights reserved.

How Total Recall Works

History data is captured from undo (and buffer cache) by the `fbda` background process at self-tuned intervals. The default is every five minutes. The entire base table row that is updated is stored, no matter how many columns are updated.

- History data is compressed using OLTP Table compression, not Hybrid Columnar compression.
Note: If the base table is compressed with Hybrid Columnar compression, the table cannot be enabled for Flashback Data Archiving.
- Each flashback archive partition is at least 1 day and 1 MB of data, partitioned on `ENDSCN`. Flashback queries to the archives avoid unrelated partitions.
- Up to ten flashback archiver slaves can be called upon by the `fbda` process.
- If the flashback archive process and slaves are too busy, archiving may be performed inline, which significantly affects the user's response time.

Oracle Total Recall Scenario

Using Flashback Data Archive to access historical data:

```
-- create the Flashback Data Archive
CREATE FLASHBACK ARCHIVE DEFAULT fla1
    TABLESPACE tbs1 QUOTA 10G RETENTION 5 YEAR;
```

①

```
-- Specify the default Flashback Data Archive
ALTER FLASHBACK ARCHIVE fla1 SET DEFAULT;
```

②

```
-- Enable Flashback Data Archive
ALTER TABLE inventory FLASHBACK ARCHIVE;
ALTER TABLE stock_data FLASHBACK ARCHIVE;
```

③

```
SELECT product_number, product_name, count FROM inventory AS
    OF TIMESTAMP TO_TIMESTAMP ('2007-01-01 00:00:00', 'YYYY-MM-
    DD HH24:MI:SS');
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Total Recall Scenario

You create a Flashback Data Archive with the `CREATE FLASHBACK ARCHIVE` statement.

- You can optionally specify the default Flashback Data Archive for the system.
- You need to provide the name of the Flashback Data Archive.
- You need to provide the name of the first tablespace of the Flashback Data Archive.
- You can identify the maximum amount of space that the Flashback Data Archive can use in the tablespace. The default is unlimited. Unless your space quota on the first tablespace is unlimited, you must specify this value, or else an ORA-55621 will ensue.
- You need to provide the retention time (number of days that Flashback Data Archive data for the table is guaranteed to be stored).

In the first example shown in the slide, a default Flashback Data Archive named `fla1` is created that uses up to 10 GB of the `tbs1` tablespace, whose data will be retained for five years. In the second example, the default Flashback Data Archive is specified. By default, the system has no Flashback Data Archive. You can set it in one of two ways:

- Specify the name of an existing Flashback Data Archive in the `SET DEFAULT` clause of the `ALTER FLASHBACK ARCHIVE` statement.
- Include `DEFAULT` in the `CREATE FLASHBACK ARCHIVE` statement when you create a Flashback Data Archive.

In the third example, Flashback Data Archive is enabled. If Automatic Undo Management is disabled, you receive an ORA-55614 if you try to modify the table.

Oracle Total Recall Scenario

Optionally, adding space:

```
ALTER FLASHBACK ARCHIVE fla1  
ADD TABLESPACE tbs3 QUOTA 5G;
```

4

Optionally, changing retention time:

```
ALTER FLASHBACK ARCHIVE fla1 MODIFY RETENTION 2 YEAR;
```

5

Optionally, purging data:

```
ALTER FLASHBACK ARCHIVE fla1 PURGE BEFORE TIMESTAMP  
(SYSTIMESTAMP - INTERVAL '1' day);
```

6

Optionally, dropping a Flashback Data Archive:

```
DROP FLASHBACK ARCHIVE fla1;
```

7

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Total Recall Scenario (continued)

To enable flashback archiving for a table, include the `FLASHBACK ARCHIVE` clause in either the `CREATE TABLE` or `ALTER TABLE` statement. In the `FLASHBACK ARCHIVE` clause, you can specify the Flashback Data Archive where the historical data for the table will be stored. The default is the default Flashback Data Archive for the system. To disable flashback archiving for a table, specify `NO FLASHBACK ARCHIVE` in the `ALTER TABLE` statement.

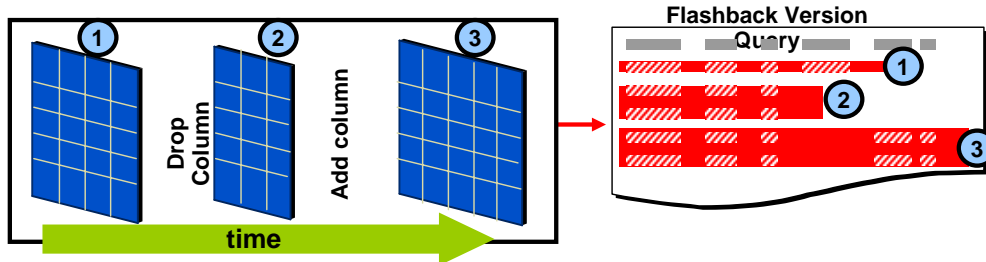
The last statement shown in the previous slide shows how to retrieve the inventory of all items at the beginning of the year 2007. Continuing the previous examples:

- Example 4 adds up to 5 GB of the `tbs3` tablespace to the `fla1` Flashback Data Archive.
- Example 5 changes the retention time for the `fla1` Flashback Data Archive to two years.
- Example 6 purges all historical data older than one day from the `fla1` Flashback Data Archive. Normally, purging is done automatically on the day after your retention time expires. You can also override this for ad hoc clean-up.
- Example 7 drops the `fla1` Flashback Data Archive and historical data, but not its tablespaces. With the `ALTER FLASHBACK ARCHIVE` command, you can:
 - Change the retention time of a Flashback Data Archive
 - Purge some or all of its data
 - Add, modify, and remove tablespaces

Note: Removing all tablespaces of a Flashback Data Archive causes an error.

Transparent Schema Evolution

- DDL support for:
 - Add, drop, rename, and modify column
 - Drop and truncate partition
 - Rename and truncate table



- Flashback queries work across DDL changes.
- All other DDL is *not* automatically supported (see next slide).

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Transparent Schema Evolution

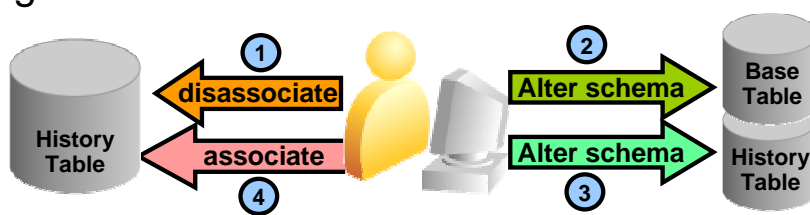
The most common DDL commands are possible with Flashback Data Archives. When a schema has evolved in any of the ways listed in the slide, Total Recall automatically keeps track of the changes. Flashback query appropriately returns the row or rows with the corresponding schema (as shown in the diagram).

Full Schema Evolution

Disassociate or associate procedures in the DBMS_FLASHBACK_ARCHIVE package:

- Disable Total Recall on specified tables and allow more complex DDL (upgrades, split tables, and so on).
- Enforce schema integrity during association. (Base table and history table must be the same schema.)

Note: This function should be used with care and with the understanding that the archive can no longer be guaranteed to be immutable because the history could have been altered during the time of disassociation.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Full Schema Evolution

All DDL changes that are not automatically supported can be executed through the DBMS_FLASHBACK_ARCHIVE package. You can use the DISASSOCIATE_FBA and REASSOCIATE_FBA procedures to disassociate and reassociate a given table from its Flashback Data Archive.

Note: This function should be used with care and with the understanding that the archive can no longer be guaranteed to be immutable, because the history could have been altered during the time of disassociation. The system catalog has a note when the disassociation occurred.

The diagram in the slide shows the following workflow:

- If you have the FLASHBACK_ARCHIVE ADMINISTER privilege, you can disassociate the archive from the base table.
- Make the necessary changes to the base table.
- Make the necessary changes to the corresponding archive.
- Then you associate the table with the archive within the same schema. Total Recall validates that the schemas are the same upon association.

Restrictions

- You cannot enable Total Recall for base tables with Hybrid Columnar compression.
- If disassociate is used, immutability of history is no longer guaranteed (but you could always purge history previously anyway with the right privilege).
- There is no transportability of history tables.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restrictions

Some DDL statements cause error ORA-55610 when used on a table enabled for Flashback Data Archive. For example:

- ALTER TABLE statement that includes an UPGRADE TABLE clause, with or without an INCLUDING DATA clause
- ALTER TABLE statement that moves or exchanges a partition or subpartition operation
- DROP TABLE statement

Guidelines

- Use SCN for precise queries.
- or*
- Use Flashback technology for your convenience.
- Flashback uses current system settings.
- Ensure database consistency with a COMMIT or ROLLBACK operation before querying past data.
- You cannot retrieve past data from a dynamic performance (V\$) view. They contain current data.
- However, you can perform queries on past data in static data dictionary views, such as *_TABLES.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Guidelines

- Use Flashback Query, Flashback Version Query, or Flashback Transaction Query for SQL code that you write, for convenience.
- Remember that all flashback processing uses the current session settings, such as national language and character set, not the settings that were in effect at the time being queried.
- To query past data at a precise time, use an SCN. If you use a time stamp, the actual time queried might be up to 3 seconds earlier than the time you specify. Oracle Database uses SCNs internally and maps them to time stamps at a granularity of 3 seconds.
- To obtain an SCN to use later with a flashback feature, you can use the DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER function.
- To compute or retrieve a past time to use in a query, use a function return value as a time-stamp or SCN argument. For example, add or subtract an INTERVAL value to the value of the SYSTIMESTAMP function.
- To ensure database consistency, always perform a COMMIT or ROLLBACK operation before querying past data.
- You cannot retrieve past data from a dynamic performance (V\$) view. A query on such a view always returns current data. However, you can perform queries on past data in static data dictionary views, such as *_TABLES.

Viewing Flashback Data Archives

Viewing the results:

View Name (DBA/USER)	Description
*_FLASHBACK_ARCHIVE	Displays information about Flashback Data Archives
*_FLASHBACK_ARCHIVE_TS	Displays tablespaces of Flashback Data Archives
*_FLASHBACK_ARCHIVE_TABLES	Displays information about tables that are enabled for flashback archiving

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Viewing Flashback Data Archives

You can use the dynamic data dictionary views to view tracked tables and Flashback Data Archive metadata. To access the `USER_FLASHBACK_*` views, you must have table ownership privileges. To inspect the `DBA_FLASHBACK_*` views, you need `SYSDBA` privileges.

For more details, see the *Advanced Application Developer's Guide* and the *PL/SQL Packages and Types Reference*.

Quiz

You cannot drop, but you can truncate, a table that is tracked by Oracle Total Recall.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Quiz

Select all correct statements about Oracle Total Recall:

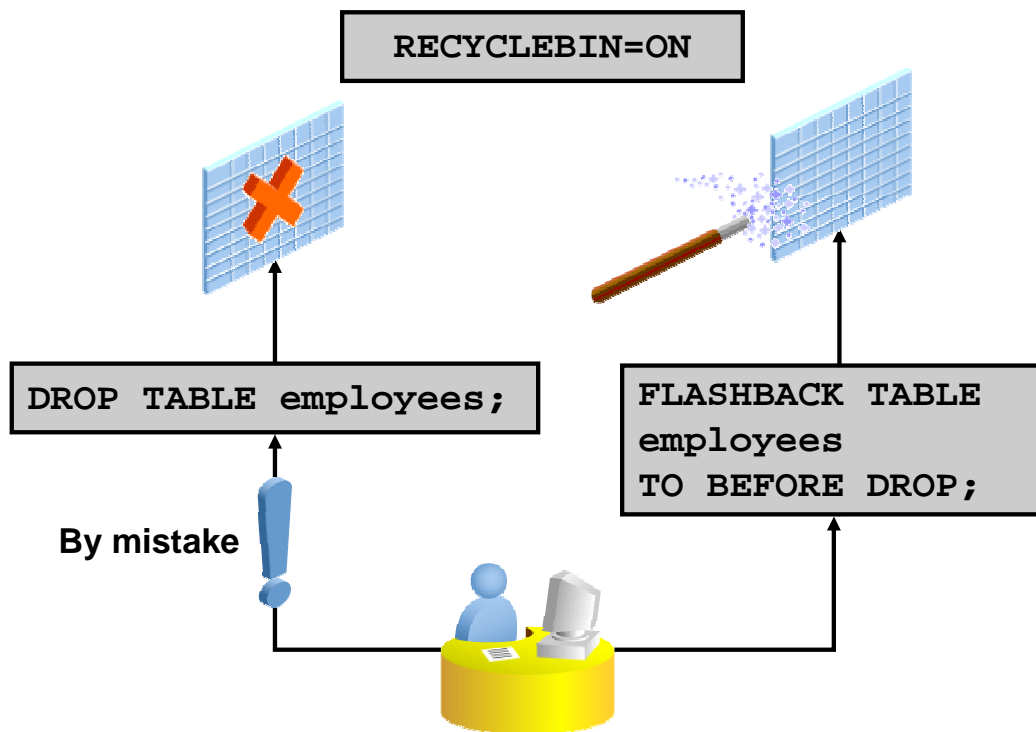
1. Oracle Total Recall is enabled by default.
2. A Flashback Data Archive provides the ability to track and store all transactional changes to a “tracked” table over its lifetime.
3. Dropping a column in a table enabled for Flashback Data Archive causes an error.
4. Flashback processing always uses the settings that were in effect at the time of being queried.
5. Flashback uses the current session settings, such as national language and character set.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 5

Flashback Drop and the Recycle Bin



ORACLE

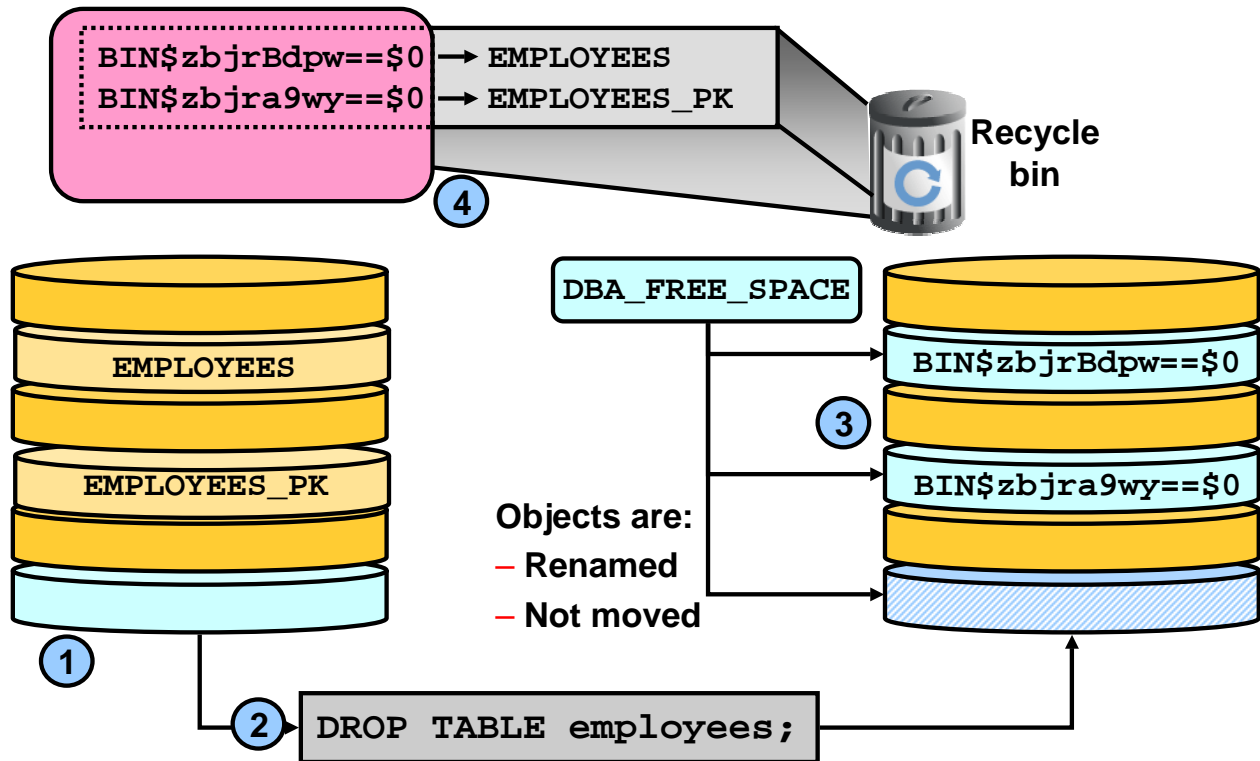
Copyright © 2009, Oracle. All rights reserved.

Flashback Drop and the Recycle Bin

Using the `FLASHBACK TABLE` command, you can undo the effects of a `DROP TABLE` statement without having to use point-in-time recovery.

Note: The `RECYCLEBIN` initialization parameter is used to control whether the Flashback Drop capability is turned ON or OFF. If the parameter is set to OFF, then dropped tables do not go into the recycle bin. If this parameter is set to ON, the dropped tables go into the recycle bin and can be recovered. By default, `RECYCLEBIN` is set to ON.

Recycle Bin



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recycle Bin

Without the recycle bin enabled, when you drop a table, the space associated with the table and its dependent objects is immediately reclaimable (that is, it can be used for other objects).

If the recycle bin is enabled, when you drop a table, then the space associated with the table and its dependent objects is not immediately reclaimable, even though it does appear in `DBA_FREE_SPACE`. Instead, the dropped objects are referenced in the recycle bin and still belong to their owner. The space used by recycle bin objects is never automatically reclaimed unless there is space pressure. This enables you to recover recycle bin objects for the maximum possible duration.

When a dropped table is “moved” to the recycle bin, the table and its associated objects and constraints are renamed using system-generated names. The renaming convention is as follows:

`BIN$unique_id$version`

where `unique_id` is a 26-character globally unique identifier for this object making the recycle bin name unique across all databases and `version` is a version number assigned by the database.

Recycle Bin (continued)

The recycle bin itself is a data dictionary table that maintains the relationships between the original names of dropped objects and their system-generated names. You can query the recycle bin by using the `DBA_RECYCLEBIN` view. The diagram in the previous slide illustrates this behavior:

1. You have created a table called `EMPLOYEES` in your tablespace.
2. You drop the `EMPLOYEES` table.
3. The extents occupied by `EMPLOYEES` are now considered as free space.
4. `EMPLOYEES` is renamed and the new name is recorded into the recycle bin.

Restoring Tables from the Recycle Bin

- Restore dropped tables and dependent objects.
- If multiple recycle bin entries have the same original name:
 - Use unique, system-generated names to restore a particular version
 - When using original names, the restored table is last in, first out (LIFO)
- Rename the original name if that name is currently used.

```
FLASHBACK TABLE <table_name> TO BEFORE DROP  
[RENAME TO <new_name>];
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restoring Tables from the Recycle Bin

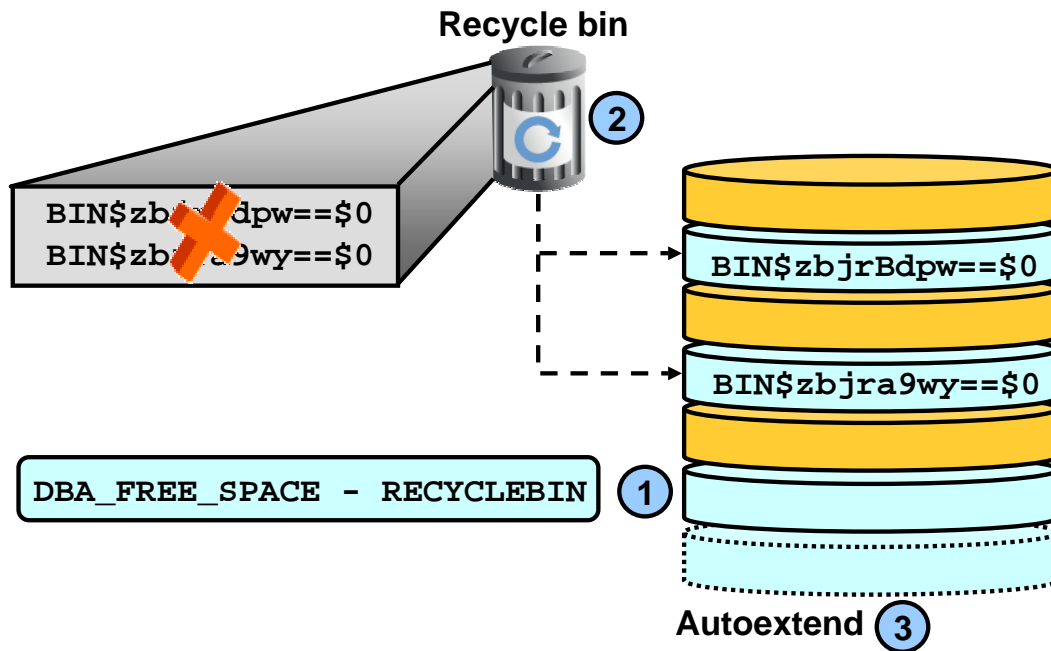
Use the `FLASHBACK TABLE . . . TO BEFORE DROP` command to recover a table and all of its possible dependent objects from the recycle bin. You can specify either the original name of the table or the system-generated name assigned to the object when it was dropped.

If you specify the original name, and if the recycle bin contains more than one object of that name, then the object that was moved to the recycle bin most recently is recovered first (LIFO: last in, first out). If you want to retrieve an older version of the table, you can specify the system-generated name of the table that you want to retrieve, or issue additional `FLASHBACK TABLE . . . TO BEFORE DROP` statements until you retrieve the table you want.

If a new table of the same name has been created in the same schema since the original table was dropped, then an error is returned unless you also specify the `RENAME TO` clause.

Note: When you flash back a dropped table, the recovered indexes, triggers, and constraints keep their recycle bin names. Therefore, it is advisable to query the recycle bin and `DBA_CONSTRAINTS` before flashing back a dropped table. In this way, you can rename the recovered indexes, triggers, and constraints to more usable names.

Recycle Bin: Automatic Space Reclamation



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recycle Bin: Automatic Space Reclamation

As long as the space used by recycle bin objects is not reclaimed, you can recover those objects by using Flashback Drop. The following are recycle bin object reclamation policies:

- Manual cleanup when you explicitly issue a PURGE command
- Automatic cleanup under space pressure: While objects are in the recycle bin, their corresponding space is also reported in DBA_FREE_SPACE because their space is automatically reclaimable. The free space in a particular tablespace is then consumed in the following order:
 1. Free space not corresponding to recycle bin objects
 2. Free space corresponding to recycle bin objects. In this case, recycle bin objects are automatically purged from the recycle bin using a first in, first out (FIFO) algorithm.
 3. Free space automatically allocated if the tablespace is auto-extensible. Suppose that you create a new table inside the TBS1 tablespace. If there is free space allocated to this tablespace that does not correspond to a recycle bin object, this free space is used as a first step. If this is not enough, free space is used that corresponds to recycle bin objects that reside inside TBS1. If the free space of some recycle bin objects is used, these objects are purged automatically from the recycle bin. At this time, you can no longer recover these objects by using the Flashback Drop feature. As a last resort, the TBS1 tablespace is extended (if possible) if the space requirement is not yet satisfied.

Recycle Bin: Manual Space Reclamation

```
PURGE {TABLE <table_name> | INDEX <index_name>}
```

```
PURGE TABLESPACE <ts_name> [USER <user_name>]
```

```
PURGE [USER_ | DBA_]RECYCLEBIN
```

ORACLE Enterprise Manager 11g Database Control

Database Instance: orcl > Tables > Recycle Bin

When you drop a table from a non-system, locally managed tablespace, Oracle does not immediately reclaim the space associated with the table. Oracle places the table and any associated objects in the Recycle Bin, where, in case the table was dropped in error, it can be recovered (Flashback Drop) at a later time.

Search

Schema Name: HR Table: Go

Results

Purge Flashback Drop

Select All Select None Expand All Collapse All

Select	Object Name	Schema	Recovery Scope	Tablespace	Drop Time	Create Time	Size	Operation
<input type="checkbox"/>	Recycle Bin							View Content
<input checked="" type="checkbox"/>	EMPLOYEE52	HR	TABLE	USERS	2007-07-02:15:45:13	2007-07-02:15:44:50.8		View Content

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recycle Bin: Manual Space Reclamation

Use the PURGE command to permanently remove objects from the recycle bin. When an object is purged from the recycle bin, the object and dependent objects are permanently removed from the database. As a consequence, objects purged from the recycle bin are no longer recoverable by using Flashback Drop. The following are possible uses of PURGE:

- PURGE TABLE purges the specified table.
- PURGE INDEX purges the specified index.
- PURGE TABLESPACE purges all the objects residing in the specified tablespace. In addition, objects residing in other tablespaces may get purged if they are dependent.
- PURGE RECYCLEBIN purges all the objects that belong to the current user. RECYCLEBIN and USER_RECYCLEBIN are synonymous.
- PURGE DBA_RECYCLEBIN purges all the objects. You must have enough system privileges or the SYSDBA system privilege to issue this command.

Tables can also be purged from the recycle bin using Enterprise Manager. On the Schema folder tab, click Tables, then select the schema the dropped object resided in and click the Recycle Bin button. Select the table from the results list and click the Purge button.

Note: For PURGE TABLE and PURGE INDEX commands, if you specify an original name and if the recycle bin contains more than one object of that name, then the object that has been in the recycle bin the longest is purged first (FIFO).

Bypassing the Recycle Bin

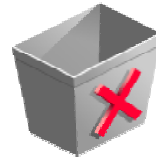
```
DROP TABLE <table_name> [PURGE] ;
```

```
DROP TABLESPACE <ts_name>  
[INCLUDING CONTENTS] ;
```

```
DROP USER <user_name> [CASCADE] ;
```

Security considerations for the recycle bin:

```
ALTER SYSTEM SET RECYCLEBIN=OFF SCOPE=SPFILE;
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Bypassing the Recycle Bin

You can use the DROP TABLE PURGE command to permanently drop a table and its dependent objects from the database. When you use this command, the corresponding objects are not moved to the recycle bin. This command provides the same functionality that the DROP TABLE command provided in previous releases.

When you issue the DROP TABLESPACE . . . INCLUDING CONTENTS command, the objects in the tablespace are not placed in the recycle bin. Moreover, objects in the recycle bin belonging to the tablespace are purged. When you issue the same command without the INCLUDING CONTENTS clause, the tablespace must be empty for the command to succeed. However, there can be objects belonging to the tablespace in the recycle bin. In this case, these objects are purged.

When you issue the DROP USER . . . CASCADE command, the user and all the objects owned by the user are permanently dropped from the database. Any objects in the recycle bin belonging to the dropped user are purged.

For increased security, you may decide to not allow the use of the recycle bin. Connected as SYSDBA, you can:

- View the recycle bin status with:
SHOW PARAMETER RECYCLEBIN
- Disable the use of the recycle bin with:
ALTER SYSTEM SET RECYCLEBIN=OFF SCOPE=SPFILE;

After issuing this command, you need to restart the database.

Querying the Recycle Bin

```
SELECT owner, original_name, object_name,  
       type, ts_name, droptime, related, space  
FROM dba_recyclebin  
WHERE can_undrop = 'YES';
```

```
SQL> SELECT original_name, object_name, ts_name, droptime  
FROM user_recyclebin WHERE can_undrop = 'YES';
```

ORIGINAL_NAME	OBJECT_NAME	TS_NAM	DROPTIME
EMPLOYEES2	BIN\$NE4Rk64w...gbpQ==\$0	USERS	2007-07-02:15:45:13

```
SQL> SHOW RECYCLEBIN
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Querying the Recycle Bin

You can view all the objects that you have dropped by querying `user_recyclebin` or `RECYCLEBIN`. It has a synonym `RECYCLEBIN`, for ease of use.

The `dba_recyclebin` view shows you all the objects that have been dropped by all users and that are still in the recycle bin.

You can also use the SQL*Plus `SHOW RECYCLEBIN` command. This command shows you only those objects that can be “undropped.”

The examples show how to extract important information from the recycle bin:

- `original_name` is the name of the object before it is dropped.
- `object_name` is the system-generated name of the object after it is dropped.
- `type` is the object’s type.
- `ts_name` is the name of the tablespace to which the object belongs.
- `droptime` is the date at which the object was dropped.
- `related` is the object identifier of the dropped object.
- `space` is the number of blocks currently used by the object.

You can also see the content of the recycle bin by using Database Control.

Note: For detailed information about the `DBA_RECYCLEBIN` view, see the *Oracle Database Reference* guide.

Quiz

When you flash back a dropped table, the recovered indexes, triggers, and constraints keep their recycle bin names.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Summary

In this lesson, you should have learned how to:

- Set up and use Total Recall
- Restore dropped tables from the recycle bin
- Query the recycle bin

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 11 Overview: Using Flashback Technology

This practice covers the following topics:

- Using Total Recall
- Recycle bin activities (*optional*)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

12

Performing Flashback Database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Configure Flashback Database
- Perform Flashback Database operations
- Monitor Flashback Database

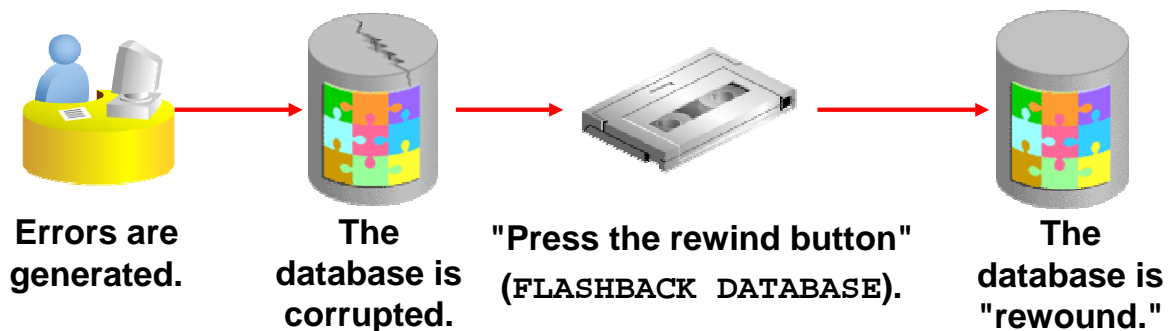
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Database

The Flashback Database operation:

- Works like a rewind button for the database
- Can be used in cases of logical data corruptions made by users



ORACLE

Copyright © 2009, Oracle. All rights reserved.

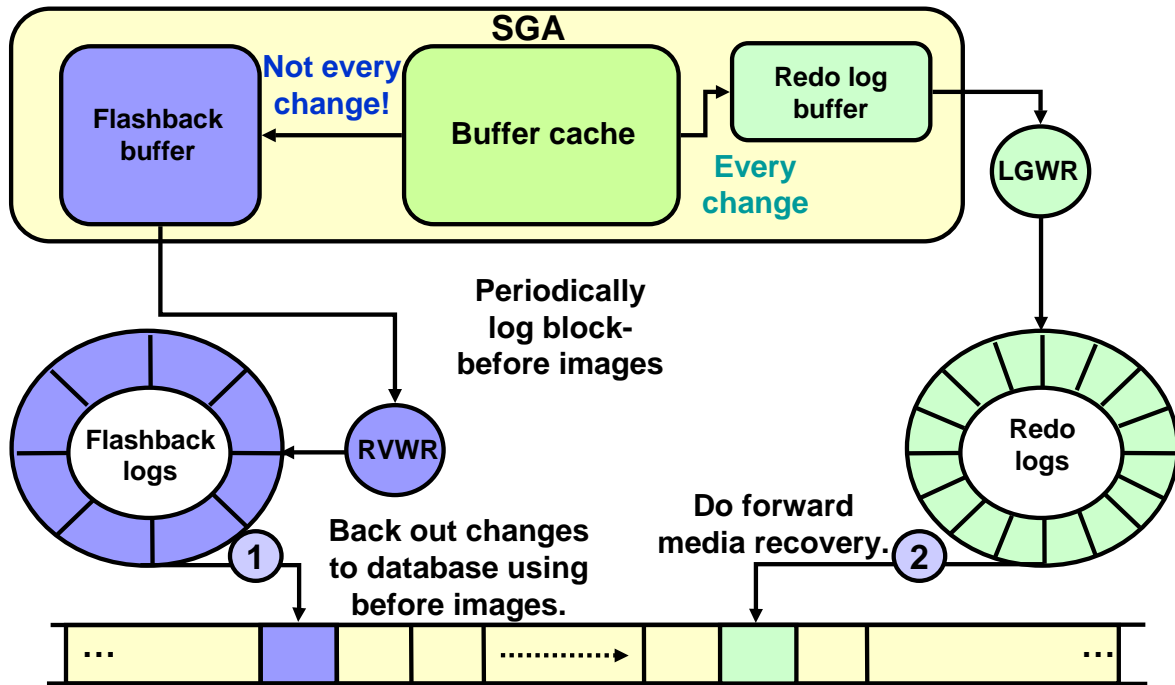
Flashback Database

With Flashback Database, you can quickly bring your database to an earlier point in time by undoing all the changes that have taken place since that time. This operation is fast because you do not need to restore backups. You can use this feature to undo changes that have resulted in logical data corruptions.

When you use Flashback Database, the Oracle database uses past block images to back out changes to the database. During normal database operation, the Oracle database occasionally logs these block images in flashback logs. Flashback logs are written sequentially and are not archived. The Oracle database automatically creates, deletes, and resizes flashback logs in the Fast Recovery Area. You need to be aware of flashback logs only for monitoring performance and deciding how much disk space to allocate for them in the Fast Recovery Area.

The time it takes to rewind a database with Flashback Database is proportional to how far back in time you need to go and the amount of database activity after the target time. The time it would take to restore and recover the whole database could be much longer. The before images in the flashback logs are used only to restore the database to a point in the past, and forward recovery is used to bring the database to a consistent state at some time in the past. The Oracle database returns data files to the previous point in time, but not auxiliary files, such as initialization parameter files. Flashback Database can also be used to compliment Data Guard and Recovery Advisor, and for synchronizing duplicated databases.

Flashback Database Architecture



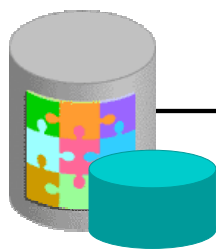
Copyright © 2009, Oracle. All rights reserved.

Flashback Database Architecture

When you enable Flashback Database, the RVWR (Flashback Writer) background process is started. This background process sequentially writes Flashback Database data from the flashback buffer to the Flashback Database logs, which are circularly reused. Subsequently, when a `FLASHBACK DATABASE` command is issued, the flashback logs are used to restore to the blocks' before images, and then redo data is used to roll forward to the desired flashback time.

The overhead of enabling Flashback Database depends on the read/write mix of the database workload. Because queries do not need to log any flashback data, the more write-intensive the workload, the higher the overhead of turning on Flashback Database.

Configuring Flashback Database



1. Configure the FRA.



2. Set the retention target.



3. Enable Flashback Database.

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP MOUNT
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER SYSTEM SET
      2 DB_FLASHBACK_RETENTION_TARGET=2880 SCOPE=BOTH;
SQL> ALTER DATABASE FLASHBACK ON;
SQL> ALTER DATABASE OPEN;
```

If your database is in ARCHIVELOG mode, there is no need to restart it.

With open database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Configuring Flashback Database

You can configure Flashback Database as follows:

1. Configure the Fast Recovery Area.
2. Set the retention target with the DB_FLASHBACK_RETENTION_TARGET initialization parameter. You can specify an upper limit, in minutes, on how far back you want to be able to flash back the database. The example uses 2,880 minutes, which is equivalent to two days. This parameter is only a target and does not provide any guarantee. Your flashback time interval depends on how much flashback data has been kept in the Fast Recovery Area.
3. Enable Flashback Database with the following command:

```
ALTER DATABASE FLASHBACK ON;
```

Before you can issue the command to enable Flashback Database, the database must be configured for archiving.

You can determine whether Flashback Database is enabled with the following query:

```
SELECT flashback_on FROM v$database;
```

You can disable Flashback Database with the ALTER DATABASE FLASHBACK OFF command. As a result, all existing Flashback Database logs are deleted automatically.

Note: You can enable Flashback Database only when the database is mounted in exclusive mode, not open.

What You Need to Do

Configuration work flow:

1. Make sure that the database is in ARCHIVELOG mode.
2. Enable flashback logging and specify the Fast Recovery Area.

Flash Recovery

This database is using a flash recovery area. The chart shows space used by each file type that is not reclaimable by Oracle. Performing backups to tertiary storage is one way to make space reclaimable. Usable Flash Recovery Area includes free and reclaimable space.

Flash Recovery Area Location

Flash Recovery Area Size GB

Flash Recovery Area Size must be set when the location is set.

Non-reclaimable Flash Recovery Area (GB)	2.07
Reclaimable Flash Recovery Area (GB)	1.38
Free Flash Recovery Area (GB)	6.55

☒ Enable Flashback Database*

Flashback database can be used for fast database point-in-time recovery, as it returns the database to a prior point-in-time without restoring files. Flashback is the preferred point-in-time recovery method in the recovery wizard when appropriate. The flash recovery area must be set to enable flashback database.

Flashback Retention Time Hours

Current size of the flashback logs(GB) n/a

Lowest SCN in the flashback data n/a

Flashback Time n/a

ORACLE

Copyright © 2009, Oracle. All rights reserved.

What You Need to Do

Log in to Enterprise Manager (EM). On the Availability page, select Recovery Settings in the Backup/Recovery region. Make sure that your database is in ARCHIVELOG mode. If not, select ARCHIVELOG Mode and then click Continue. You need to shut down and restart the instance for your changes to take effect.

When the Fast Recovery Area and archiving are configured, `USE_DB_RECOVERY_FILE_DEST` is configured for archive log destination 10. Enable flashback logging by selecting Enable Flashback Logging. You can also set the flashback retention time and view important information regarding your flashback database window.

Review the Fast Recovery Area location. The Fast Recovery Area is a unified storage location for all recovery-related files and activities in an Oracle database. All files that are needed to completely recover a database from a media failure are part of the Fast Recovery Area. The recovery-related files that can be created in the Fast Recovery Area include: archived redo log files, control files, backups created by Recovery Manager (RMAN), flashback logs, and the change tracking file. By allocating a storage location and unifying recovery-related files within a specific area, the Oracle database server relieves the database administrator from having to manage the disk files created by these components. The default location for the Fast Recovery Area is `$ORACLE_BASE/flash_recovery_area`. If you would like it in a different location, change it now. Scroll down to the bottom of the Recovery Settings page and click Apply.

Flashback Database: Examples

- To flash back: Mounted (in exclusive mode) database

```
RMAN> FLASHBACK DATABASE TO TIME =  
2> "TO_DATE('2009-05-27 16:00:00',  
3> 'YYYY-MM-DD HH24:MI:SS')";
```

```
RMAN> FLASHBACK DATABASE TO SCN=23565;  
RMAN> FLASHBACK DATABASE  
2> TO SEQUENCE=223 THREAD=1;
```

Monitor progress of Flashback Database with the V\$SESSION_LONGOPS view.

```
SQL> FLASHBACK DATABASE  
2 TO TIMESTAMP(SYSDATE-1/24);  
SQL> FLASHBACK DATABASE TO SCN 53943;  
SQL> FLASHBACK DATABASE TO RESTORE POINT b4_load;
```

- To review changes: Read-only opened database
- To finalize: Read/write opened database with RESETLOGS

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Database: Examples

You can use the RMAN FLASHBACK DATABASE command to execute the Flashback Database operation. You can use SEQUENCE and THREAD to specify a redo log sequence number and thread as a lower limit. RMAN selects only files that can be used to flash back to, but not including, the specified sequence number.

Alternatively, you can use the SQL FLASHBACK DATABASE command to return the database to a past time or SCN. If you use the TO SCN clause, you must provide a number. If you specify TO TIMESTAMP, you must provide a time stamp value. You can also specify a restore point name.

You can monitor the Flashback Database progress with the V\$SESSION_LONGOPS view.

Note: The database must be mounted in exclusive mode to issue the FLASHBACK DATABASE command and opened read-only to review changes. The database must be opened read/write with the RESETLOGS option when finished.

Flashback Database Considerations

- When the Flashback Database operation completes, open the database:
 - In read-only mode to verify that the correct target time or SCN was used
 - With a `RESETLOGS` operation to allow DML
- The opposite of “flash back” is “recover.”
- You cannot use Flashback Database in the following situations:
 - The control file has been restored or re-created.
 - A tablespace has been dropped.
 - A data file has been reduced in size.
- Use the `TO BEFORE RESETLOGS` clause to flash back to before the last `RESETLOGS` operation.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Database Considerations

In situations where you cannot use the Flashback Database feature, you should use an incomplete recovery operation to return the database to a specific time. After the Flashback Database operation is complete, you can open the database in read-only mode to verify that the correct target time or SCN was used. If not, you can flash back the database again, or perform a recovery to roll forward the database. So, to undo a Flashback Database operation, you should recover the database forward.

You cannot use Flashback Database to recover a data file that was dropped during the span of time you are flashing back. The dropped data file is added to the control file and marked offline, but it is not flashed back. Flashback Database cannot flash back a data file to a time after its creation and before the resize operation. If a file was resized during the span of time to which you are going to flash back the database, then you should take the file offline before beginning the Flashback Database operation. This is applicable for files that are shrunk rather than expanded. You can use Flashback Database with data files that you have configured for automatic extension. You can flash back to just before the last `RESETLOGS` operation by supplying the `TO BEFORE RESETLOGS` clause in the `FLASHBACK DATABASE` command.

Note: The flashback retention target is not an absolute guarantee that flashback will be available. If space is needed for required files in the Fast Recovery Area, flashback logs may be deleted automatically.

Monitoring Flashback Database

To monitor the ability to meet your retention target:

- View the Fast Recovery Area disk quota:

```
SQL> SELECT estimated_flashback_size,  
2         flashback_size  
3 FROM    V$FLASHBACK_DATABASE_LOG;
```

- Determine the current flashback window:

```
SQL> SELECT oldest_flashback_scn,  
2         oldest_flashback_time  
3 FROM      V$FLASHBACK_DATABASE_LOG;
```

- Monitor logging in the Flashback Database logs:

```
SQL> SELECT *  
2 FROM      V$FLASHBACK_DATABASE_STAT;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring Flashback Database

It is important for you to monitor space usage of the Fast Recovery Area so that you know how well you are meeting your retention target. Use the V\$FLASHBACK_DATABASE_LOG view to monitor the Flashback Database retention target:

- ESTIMATED_FLASHBACK_SIZE uses previously logged flashback data to provide an estimate of how much disk space is needed in the Fast Recovery Area for flashback logs to meet the current flashback retention target. The estimate is based on the workload since the instance was started, or during the most recent time interval equal to the flashback retention target, whichever is shorter.
- FLASHBACK_SIZE gives you the current size, in bytes, of the flashback data.
- OLDEST_FLASHBACK_SCN and OLDEST_FLASHBACK_TIME display the approximate lowest SCN and time to which you can flash back your database. CURRENT_SCN in V\$DATABASE gives you the current database SCN.

Use the V\$FLASHBACK_DATABASE_STAT view to monitor the overhead of logging flashback data in the Flashback Database logs. This view contains 24 hours of information, with each row representing a one-hour time interval. You can use this view to determine rate changes in the flashback data generation.

```
SQL> SELECT begin_time, end_time, flashback_data, db_data,  
2         redo_data, estimated_flashback_size AS EST_FB_SIZE  
3 FROM      V$FLASHBACK_DATABASE_STAT;
```

Monitoring Flashback Database (continued)

BEGIN_TIME	END_TIME	FLASHBACK_DATA	DB_DATA	REDO_DATA	EST_FB_SIZE
12-FEB-09	12-FEB-09	16384	0	24576	0
12-FEB-09	12-FEB-09	6594560	7471104	1533440	815923200
12-FEB-09	12-FEB-09	17235968	12361728	5150920	839467008
12-FEB-09	12-FEB-09	311648256	37249024	10272768	855195648

Based on this information, you may need to adjust the retention time or the Fast Recovery Area size.

FLASHBACK_DATA and REDO_DATA represent the number of bytes of flashback data and redo data written, respectively, during the time interval, and DB_DATA gives the number of bytes of data blocks read and written. This view also contains the estimated flashback space needed for the interval.

You can query V\$RECOVERY_FILE_DEST to view information regarding the Fast Recovery Area. The column descriptions are:

- **NAME:** Fast Recovery Area name, indicating location string
- **SPACE_LIMIT:** Disk limit specified in the DB_RECOVERY_FILE_DEST_SIZE parameter
- **SPACE_USED:** Space used by Fast Recovery Area files (in bytes)
- **SPACE_RECLAIMABLE:** Amount of space that can be reclaimed by deleting obsolete, redundant, and other low-priority files through the space management algorithm
- **NUMBER_OF_FILES:** Number of files

```
SQL> SELECT name, space_limit AS quota,
2          space_used          AS used,
3          space_reclaimable AS reclaimable,
4          number_of_files    AS files
5 FROM v$recovery_file_dest ;
```

NAME	QUOTA	USED	RECLAIMABLE	FILES
/u01/flash_recovery_area	5368707120	2507809104	203386880	226

Monitoring Flashback Database with EM

Flash Recovery

This database is using a flash recovery area. The chart shows space used by each file type that is not reclaimable by Oracle. Performing backups to tertiary storage is one way to make space reclaimable. Usable Flash Recovery Area includes free and reclaimable space.

Flash Recovery Area Location

Flash Recovery Area Size

Flash Recovery Area Size must be set when the location is set.

Non-reclaimable Flash Recovery Area (MB) 294

Reclaimable Flash Recovery Area (B) 0

Free Flash Recovery Area (GB) 5.71

☐ Enable Flashback Database*

Flashback database can be used for fast database point-in-time recovery, as it returns the database to a prior point-in-time without restoring files. Flashback is the preferred point-in-time recovery method in the recovery wizard when appropriate. The flash recovery area must be set to enable flashback database.

Flashback Retention Time

Current size of the flashback logs(GB) n/a

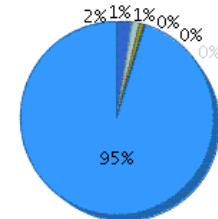
Lowest SCN in the flashback data n/a

Flashback Time n/a

☐ Apply initialization parameter changes to SPFILE only. If not checked, parameter changes will be made to both the SPFILE and the running instance.

* Changes to this setting or parameter require a database restart.

Flash Recovery Area Usage



Online Log	- 0.15GB (2.5%)
Backup Piece	- 0.08GB (1.3%)
Archived Redo Log	- 0.05GB (0.8%)
Control File	- 0.01GB (0.2%)
Image Copy	- 0GB (0%)
Flashback Log	- 0GB (0%)
Usable	- 5.71GB (95.2%)

ORACLE

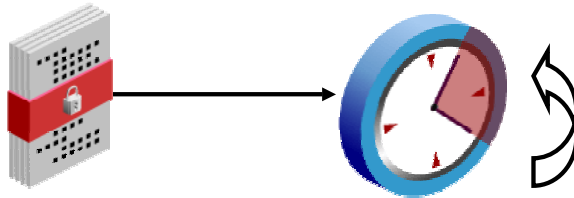
Copyright © 2009, Oracle. All rights reserved.

Monitoring Flashback Database with EM

Most of the Flashback Database statistics mentioned on the preceding pages can be viewed from the Recovery Settings page. These metrics include the current space used by all flashback logs, the lowest SCN, and the time of the lowest SCN in the flashback data.

Guaranteed Restore Points

A guaranteed restore point ensures that you can perform a FLASHBACK DATABASE command to that SCN at any time.



```
SQL> CREATE RESTORE POINT before_upgrade  
2    GUARANTEE FLASHBACK DATABASE;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Guaranteed Restore Points

Like normal restore points, guaranteed restore points can be used as aliases for SCNs in recovery operations. A principal difference is that guaranteed restore points never age out of the control file and must be explicitly dropped. However, they also provide specific functionality related to the use of the Flashback Database feature.

Creating a guaranteed restore point at a particular SCN enforces the requirement that you can perform a Flashback Database operation to return your database to its state at that SCN, even if flashback logging is not enabled for your database. If flashback logging is enabled, creating a guaranteed restore point enforces the retention of flashback logs required for Flashback Database back to any point in time after the creation of the earliest guaranteed restore point.

A guaranteed restore point can be used to revert a whole database to a known good state days or weeks ago, as long as there is enough disk space in the Fast Recovery Area to store the needed logs. As with normal restore points, guaranteed restore points can be used to specify a point in time for RECOVER DATABASE operations.

Note: Limitations that apply to Flashback Database also apply to guaranteed restore points. For example, shrinking a data file or dropping a tablespace can prevent flashing back the affected data files to the guaranteed restore point.

Flashback Database and Guaranteed Restore Points

To use guaranteed restore points, the database must satisfy the following prerequisites:

- The `COMPATIBLE` initialization parameter must be set to 10.2 or greater.
- The database must be running in `ARCHIVELOG` mode.
- `FLASHBACK DATABASE` requires the use of archived redo logs starting from around the time of the restore point.
- A Fast Recovery Area must be configured.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Flashback Database and Guaranteed Restore Points

To support the use of guaranteed restore points, the database must satisfy the following prerequisites:

- The `COMPATIBLE` initialization parameter must be set to 10.2 or greater.
- The database must be running in `ARCHIVELOG` mode.
- To rewind the database to a guaranteed restore point, the `FLASHBACK DATABASE` command needs the archived redo logs starting from around the time of the restore point.
- A Fast Recovery Area must be configured. Guaranteed restore points use a mechanism similar to flashback logging. As with flashback logging, the Oracle database must store the required logs in the Fast Recovery Area.
- If Flashback Database is not enabled, then the database must be mounted, not open, when creating the first guaranteed restore point (or if all previously created guaranteed restore points have been dropped).

Logging for Flashback Database and guaranteed restore points involves capturing images of data file blocks before changes are applied. The `FLASHBACK DATABASE` command can use these images to return the data files to their previous state. The chief differences between normal flashback logging and logging for guaranteed restore points are related to when blocks are logged and whether the logs can be deleted in response to space pressure in the Fast Recovery Area. These differences affect space usage for logs and database performance.

Flashback Database and Guaranteed Restore Points (continued)

If you enable Flashback Database and define one or more guaranteed restore points, then the database performs normal flashback logging. In this case, the recovery area retains the flashback logs required to flash back to any arbitrary time between the present and the earliest currently defined guaranteed restore point. Flashback logs are not deleted in response to space pressure if they are required to satisfy the guarantee.

Quiz

You can use Flashback Database, when you want to:

1. Repair logical data corruptions
2. Recover a tablespace that has been dropped
3. Recover to a point prior to when a data file has been reduced in size
4. Recover to a point prior to when you re-created the control file

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Quiz

Flashback logs are archived to allow you to rewind to a point in time that your FRA cannot accommodate.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Summary

In this lesson, you should have learned how to:

- Configure Flashback Database
- Perform Flashback Database operations
- Monitor Flashback Database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 12 Overview: Working with Flashback Database

This practice covers the following topics:

- Performing Flashback Database to undo unwanted transactions
- Monitoring the Flashback Database retention
- Determining the size of the flashback logs

ORACLE

Copyright © 2009, Oracle. All rights reserved.

13

Managing Memory

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the memory components in the SGA
- Implement Automatic Memory Management
- Manually configure SGA parameters
- Configure automatic PGA memory management

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Memory Management: Overview

DBAs must consider memory management to be a crucial part of their job because:

- There is a finite amount of memory available
- Allocating more memory to serve certain types of functions can improve overall performance
- Automatically tuned memory allocation is often the appropriate configuration, but specific environments or even short-term conditions may require further attention

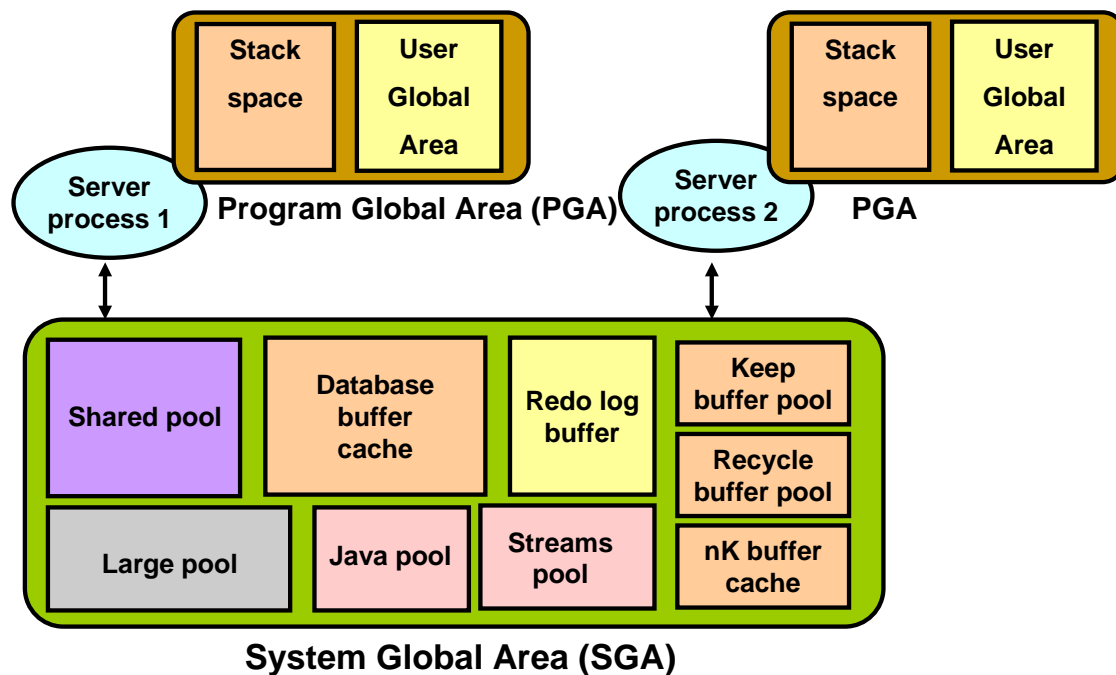
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Memory Management: Overview

Because there is a finite amount of memory available on a database server and thus, on an Oracle database instance, you must pay attention to how memory is allocated. If too much memory is allowed to be used by a particular area that does not need it, then there is the possibility that there are other functional areas unnecessarily doing without enough memory to perform optimally. With the ability to have memory allocation automatically determined and maintained for you, the task is simplified greatly. But even automatically tuned memory needs to be monitored for optimization and may need to be manually configured to some extent.

Reviewing Oracle Database Memory Structures



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Reviewing Oracle Database Memory Structures

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, that 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.

Reviewing Oracle Database Memory Structures (continued)

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

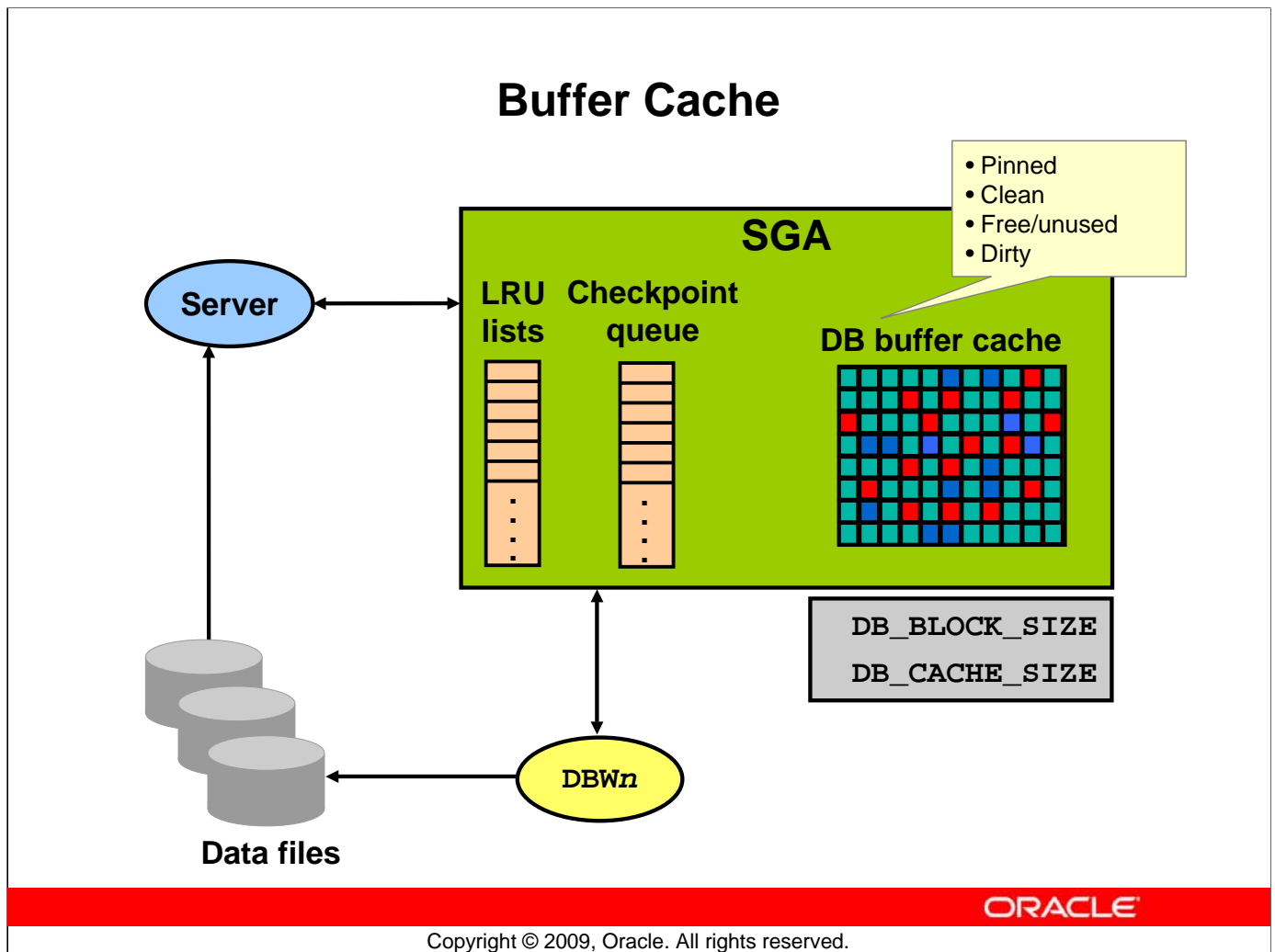
- **Shared pool:** Caches various constructs that can be shared among users
- **Database buffer cache:** Caches blocks of data retrieved from the database
- **KEEP buffer pool:** Is a specialized type of database buffer cache that is tuned to retain blocks of data in memory for long periods of time
- **Recycle buffer pool:** Is a specialized type of database buffer cache that is tuned to recycle or remove block from memory quickly
- **nK buffer cache:** Is one of several specialized database buffer caches designed to hold block sizes different from the default database block size
- **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:** Is the 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:** Is used for all session-specific Java code and data in the Java Virtual Machine (JVM)
- **Streams pool:** Is used by Oracle Streams to store information required by capture and apply

When you start the instance by using Enterprise Manager or SQL*Plus, 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 created 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 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).



Buffer Cache

You can configure the buffer cache by specifying a value for the `DB_CACHE_SIZE` parameter. The buffer cache holds copies of the data blocks from the data files having a block size of `DB_BLOCK_SIZE`. The buffer cache is a part of the SGA, so all users can share these blocks. The server processes read data from the data files into the buffer cache. To improve performance, the server process sometimes reads multiple blocks in a single read operation. The `DBWn` process writes data from the buffer cache into the data files. To improve performance, `DBWn` writes multiple blocks in a single write operation.

At any given time, the buffer cache may hold multiple copies of a single database block. Only one current copy of the block exists, but to satisfy queries, server processes may need to construct read-consistent copies from past image information. This is called a consistent read (CR) block.

The least recently used (LRU) list reflects the usage of buffers. The buffers are sorted on the basis of a combination of how recently and how often they have been referenced. Thus, buffers that are most frequently and recently used are found at the most recently used end. Incoming blocks are copied to a buffer from the least recently used end, which is then assigned to the middle of the list, as a starting point. From here, the buffer works its way up or down the list, depending on usage.

Buffer Cache (continued)

Buffers in the buffer cache can be in one of four states:

- **Pinned:** The block is either currently being read into the cache or being written to. Other sessions wait to access the block.
- **Clean:** The buffer is now unpinned and is a candidate for immediate aging out if the current contents (data block) are not referenced again. Either the contents are in sync with disk or the buffer contains a CR snapshot of a block.
- **Free/unused:** The buffer is empty because the instance just started. This state is very similar to the clean state, except that the buffer has not been used.
- **Dirty:** The buffer is no longer pinned but the contents (data block) have changed and must be flushed to disk by DBWn before it can be aged out.

Server processes use the buffers in the buffer cache, but the DBWn process makes buffers in the cache available by writing changed buffers back to the data files. The checkpoint queue lists the buffers that are to be written out to disk.

Then Oracle database supports multiple block sizes in the same database. The standard block size is used for the SYSTEM tablespace. You specify the standard block size by setting the initialization parameter DB_BLOCK_SIZE. Legitimate values are from 2 KB to 32 KB, and the default is 8 KB.

The cache sizes of nonstandard block size buffers are specified by the following parameters:

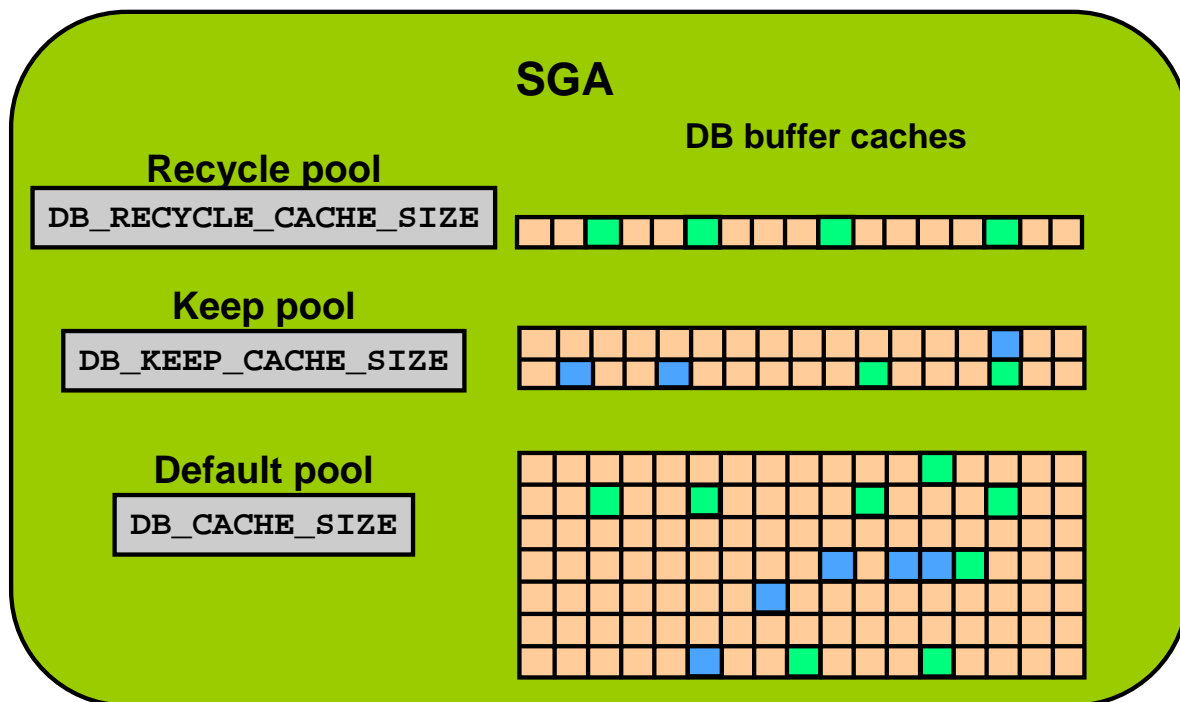
- DB_2K_CACHE_SIZE
- DB_4K_CACHE_SIZE
- DB_8K_CACHE_SIZE
- DB_16K_CACHE_SIZE
- DB_32K_CACHE_SIZE

The DB_nK_CACHE_SIZE parameters cannot be used to size the cache for the standard block size. If the value of DB_BLOCK_SIZE is nK, it is illegal to set DB_nK_CACHE_SIZE. The size of the cache for the standard block size is always determined from the value of DB_CACHE_SIZE.

Each buffer cache has a limited size, so typically not all the data on disk can fit in the cache. When the cache is full, subsequent cache misses cause the Oracle database to write dirty data already in the cache to disk to make room for the new data. (If a buffer is not dirty, it does not need to be written to disk before a new block can be read into the buffer.) Subsequent access to any data that was written to disk results in additional cache misses.

The size of the cache affects the likelihood that a request for data will result in a cache hit. If the cache is large, it is more likely to contain the data that is requested. Increasing the size of a cache increases the percentage of data requests that result in cache hits.

Using Multiple Buffer Pools



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Multiple Buffer Pools

The database administrator (DBA) may be able to improve the performance of the database buffer cache by creating multiple buffer pools. You assign objects to a buffer pool depending on how the objects are accessed. There are three buffer pools:

- **Keep:** This pool is used to retain objects in memory that are likely to be reused. Keeping these objects in memory reduces I/O operations. Buffers are kept in this pool by ensuring that the pool is sized larger than the total size of the segments assigned to the pool. This means that buffers do not have to be aged out. The keep pool is configured by specifying a value for the `DB_KEEP_CACHE_SIZE` parameter.
- **Recycle:** This pool is used for blocks in memory that have little chance of being reused. The recycle pool is sized smaller than the total size of the segments assigned to the pool. This means that blocks read into the pool will often have to age out a buffer. The recycle pool is configured by specifying a value for the `DB_RECYCLE_CACHE_SIZE` parameter.
- **Default:** This pool always exists. It is equivalent to the buffer cache of an instance without a keep pool or a recycle pool and is configured with the `DB_CACHE_SIZE` parameter.

Note: The memory in the keep or recycle pool is not a subset of the default buffer pool.

Using Multiple Buffer Pools

```
CREATE INDEX cust_idx ...  
  STORAGE (BUFFER_POOL KEEP);  
  
ALTER TABLE oe.customers  
  STORAGE (BUFFER_POOL RECYCLE);  
  
ALTER INDEX oe.cust_lname_ix  
  STORAGE (BUFFER_POOL KEEP);
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Multiple Buffer Pools (continued)

The `BUFFER_POOL` clause is used to define the default buffer pool for an object. It is part of the `STORAGE` clause and is valid for `CREATE` and `ALTER` table, cluster, and index statements. The blocks from an object without an explicitly set buffer pool go into the default buffer pool.

The syntax is `BUFFER_POOL [KEEP | RECYCLE | DEFAULT]`.

When the default buffer pool of an object is changed using the `ALTER` statement, blocks that are already cached remain in their current buffers until they are flushed out by the normal cache management activity. Blocks read from disk are placed into the newly specified buffer pool for the segment.

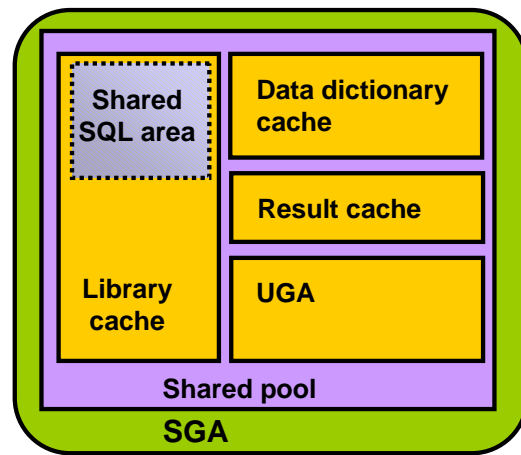
Because buffer pools are assigned to a segment, objects with multiple segments can have blocks in multiple buffer pools. For example, an index-organized table can have different pools defined on both the index and the overflow segment.

Shared Pool

Contents:

- Library cache: Command text, parsed code, and execution plan
- Data dictionary cache: Definitions for tables, columns, and privileges from the data dictionary tables
- Result cache: Results from SQL queries and PL/SQL functions
- User Global Area (UGA): Session information for the Oracle shared server

SHARED_POOL_SIZE



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Shared Pool

You can specify the size of the shared pool with the `SHARED_POOL_SIZE` initialization parameter. The shared pool is a memory area that stores information shared by multiple sessions. It contains different types of data, as shown in the graphic in the slide.

Library cache: The library cache contains shared SQL and PL/SQL areas—the fully parsed or compiled representations of PL/SQL blocks and SQL statements. PL/SQL blocks include:

- Procedures and functions
- Packages
- Triggers
- Anonymous PL/SQL blocks

Data dictionary cache: The data dictionary cache holds definitions of dictionary objects in memory.

Result cache: The result cache comprises the SQL query result cache and PL/SQL function result cache. This cache is used to store results of SQL queries or PL/SQL functions to speed up their future execution.

User Global Area: The UGA contains the session information for the Oracle shared server. The UGA is located in the shared pool when using a shared server session and if the large pool is not configured.

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
 - Parallel query operations
 - Advanced Queuing memory table storage
- Reduces potential fragmentation of shared pool
- Is managed by AMM and ASMM
- Is sized with the `LARGE_POOL_SIZE` parameter

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Large Pool

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

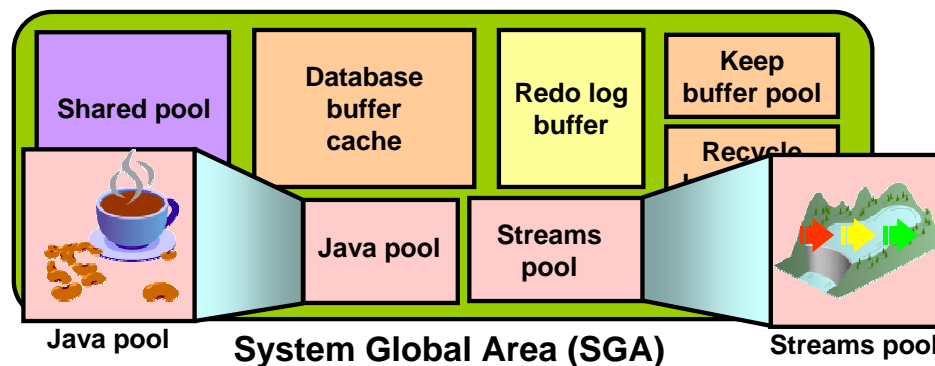
- Session memory for the shared server and the Oracle XA interface (used where transactions interact with multiple databases)
- I/O server processes
- Buffers for Recovery Manager (RMAN) I/O slaves
- Message buffers used in the parallel execution of statements
- Advanced Queuing memory table storage

By allocating session memory for the items listed in the slide, the shared pool has less fragmentation that would come from having large objects frequently allocated and deallocated in it. Segregating large objects out of the shared pool results in more efficient shared pool usage, which means more of its memory is available to service new requests and to retain existing data if needed.

The large pool can be automatically managed by AMM and ASMM. You can also size it with the `LARGE_POOL_SIZE` parameter.

Java Pool and Streams Pool

- Java pool memory is used in server memory for all session-specific Java code and data in the JVM.
- Streams pool memory is used exclusively by Oracle Streams to:
 - Store buffered queue messages
 - Provide memory for Oracle Streams processes



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Java Pool and Streams Pool

Java pool memory is used in server memory for all session-specific Java code and data in the JVM. Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

The Java Pool Advisor statistics provide information about library cache memory used for Java and predict how changes in the size of the Java pool can affect the parse rate. The Java Pool Advisor is internally turned on when `statistics_level` is set to `TYPICAL` or higher. These statistics reset when the advisor is turned off.

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.

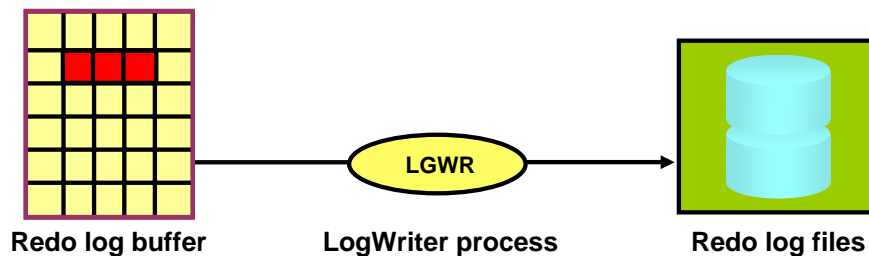
Note: A detailed discussion of Java programming and Oracle Streams is beyond the scope of this class.

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

Content transferred by log writer process (LGWR):

- When a user process commits a transaction
- When the redo log buffer is one-third full
- Before a `DBWn` process writes modified buffers to disk



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Redo Log Buffer

The Oracle server processes copy redo entries from the user's memory space to the redo log buffer for each DML or DDL statement. The redo entries contain the information necessary to reconstruct or redo changes made to the database by DML and DDL operations. They are used for database recovery and take up continuous sequential space in the buffer.

The redo log buffer is a circular buffer; the server processes can copy new entries over the entries in the redo log buffer that have already been written to disk. The LGWR process normally writes fast enough to ensure that space is always available in the buffer for new entries. The LGWR process writes the redo log buffer to the active online redo log file (or members of the active group) on disk. The LGWR process copies to disk all redo entries that have been entered into the buffer since the last time LGWR wrote to disk.

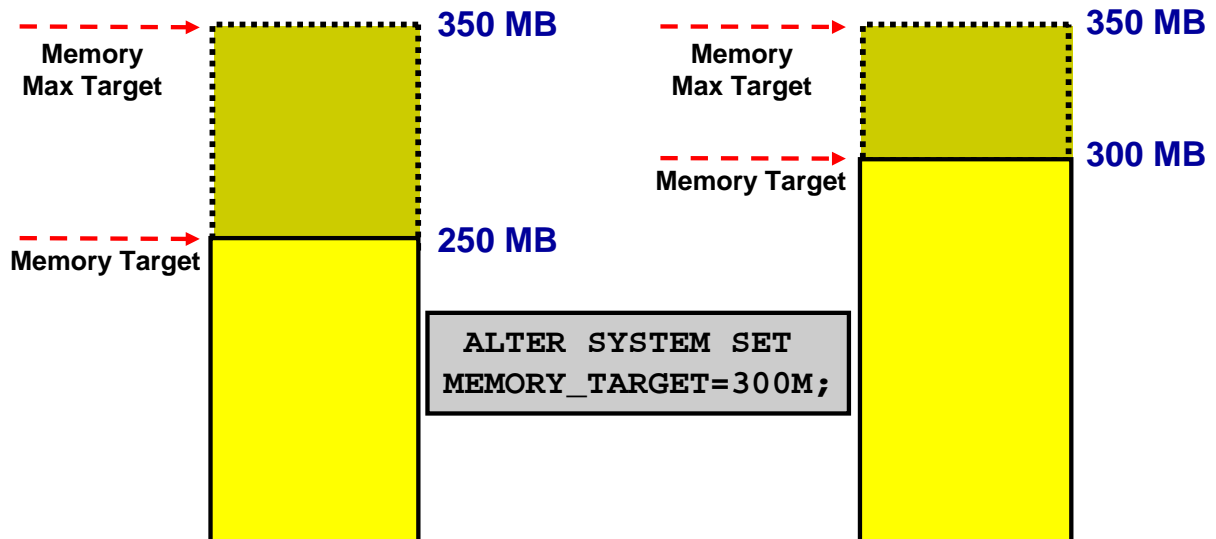
What Causes LGWR to Write?

LGWR writes out the redo data from the redo log buffer:

- When a user process commits a transaction
- Every three seconds, or when the redo log buffer is one-third full
- When a `DBWn` process writes modified buffers to disk, if the corresponding redo log data has not already been written to disk

Automatic Memory Management: Overview

With Automatic Memory Management, the database can size the SGA and PGA automatically according to your workload.



Oracle recommends the use of AMM unless you have special requirements.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Automatic Memory Management: Overview

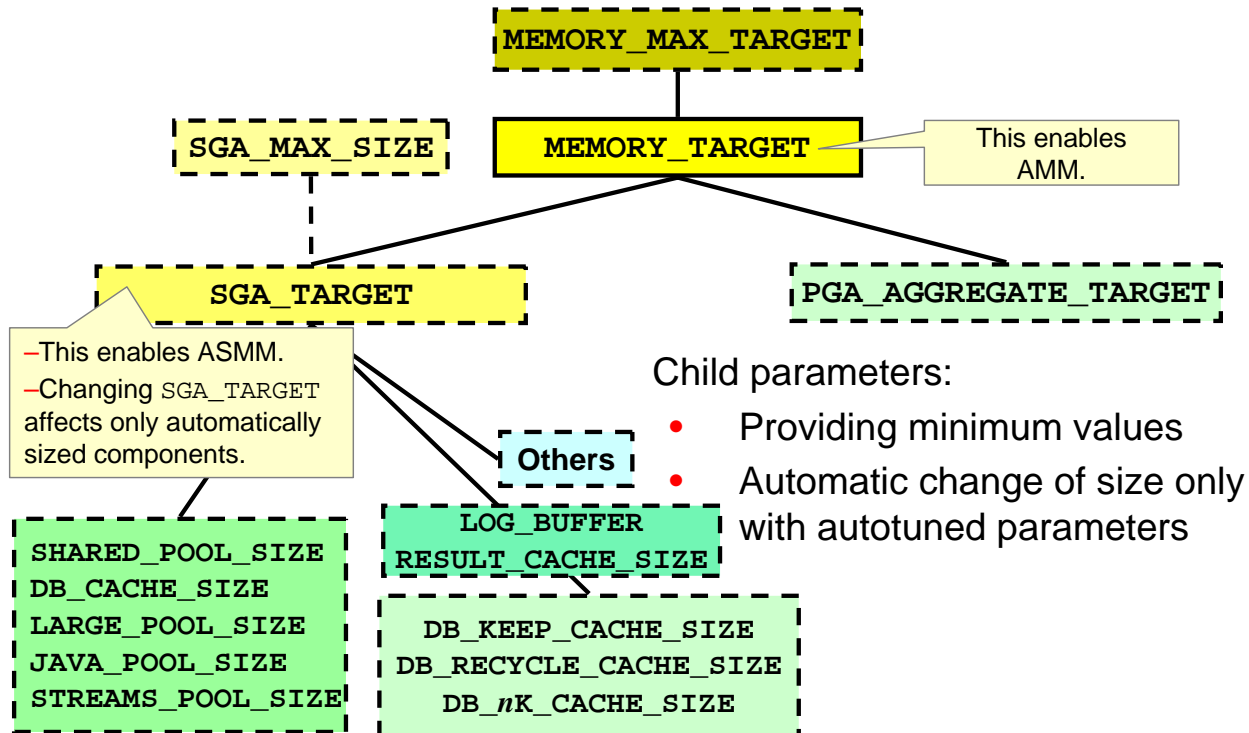
Automatic Memory Management (AMM) allows the Oracle Database to manage SGA memory and instance PGA memory sizing automatically. To do so (on most platforms), you set only a target memory size initialization parameter (`MEMORY_TARGET`) and a maximum memory size initialization parameter (`MEMORY_MAX_TARGET`), and the database dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands. You can enable AMM in Enterprise Manager by navigating to: Server > Memory Advisors (in the Database Configuration section) and then clicking the Enable button.

With this memory management method, the database also dynamically tunes the sizes of the individual SGA components and the sizes of the individual PGAs.

Because the target memory initialization parameter is dynamic, you can change the target memory size at any time without restarting the database. The maximum memory size serves as an upper limit so that you cannot accidentally set the target memory size too high. Because certain SGA components either cannot easily shrink or must remain at a minimum size, the database also prevents you from setting the target memory size too low.

This indirect memory transfer relies on the operating system (OS) mechanism of freeing shared memory. After memory is released to the OS, the other components can allocate memory by requesting memory from the OS. Currently, Automatic Memory Management is implemented on Linux, Solaris, HP-UX, AIX, and Windows.

Oracle Database Memory Parameters



ORACLE

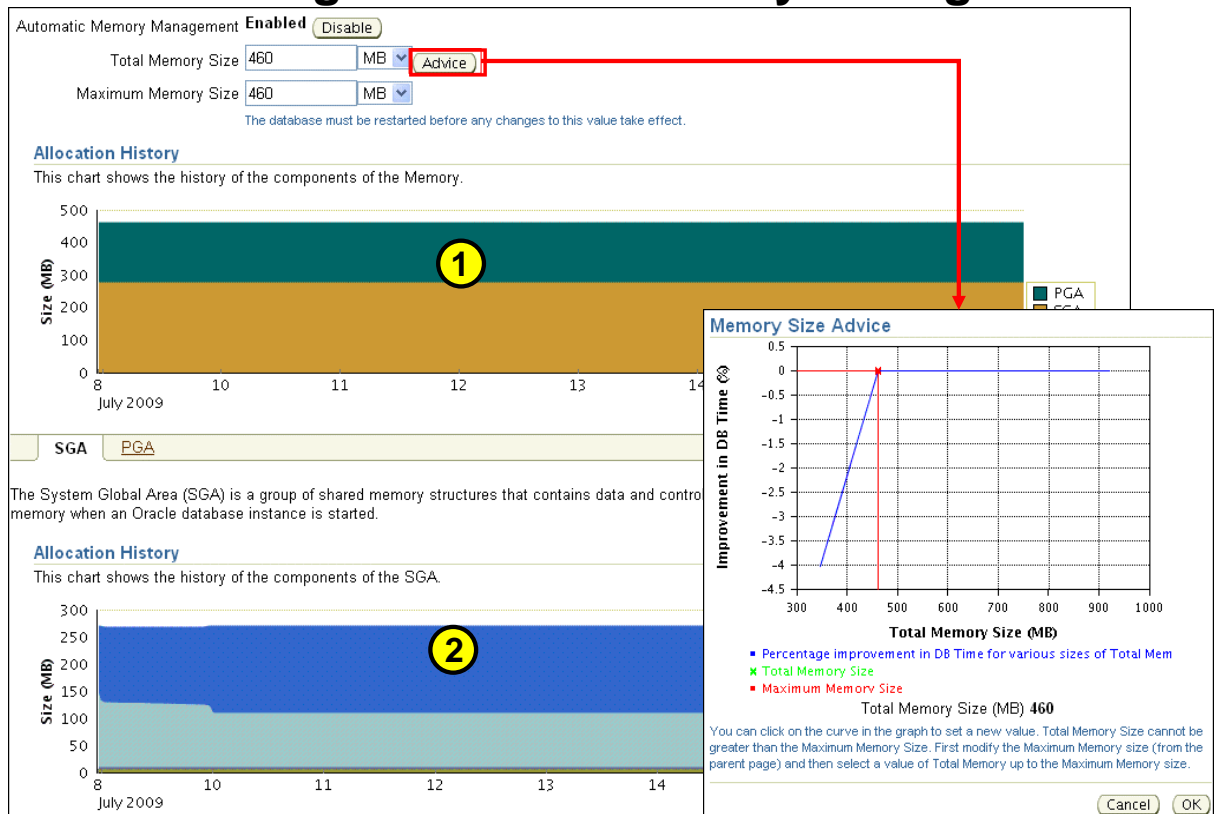
Copyright © 2009, Oracle. All rights reserved.

Oracle Database Memory Sizing Parameters

The graphic in the slide shows you the memory initialization parameters hierarchy. Although you have to set only `MEMORY_TARGET` to trigger Automatic Memory Management, you still have the possibility to set lower bound values for various caches. Therefore, if the child parameters are user set, they will be the minimum values below which the Oracle database server will not autotune that component.

- If `SGA_TARGET` and `PGA_AGGREGATE_TARGET` are set to a nonzero value, they are considered to be the minimum values for the sizes of the SGA and the PGA, respectively. `MEMORY_TARGET` can take values from `SGA_TARGET + PGA_AGGREGATE_TARGET` to `MEMORY_MAX_SIZE`.
- If `SGA_TARGET` is set, the database autotunes only the sizes of the subcomponents of the SGA. PGA is autotuned independent of whether it is explicitly set or not. However, the whole SGA (`SGA_TARGET`) and the PGA (`PGA_AGGREGATE_TARGET`) are not autotuned—that is, do not grow or shrink automatically.

Monitoring Automatic Memory Management



Copyright © 2009, Oracle. All rights reserved.

Monitoring Automatic Memory Management

From the EM home page (Related Links section), navigate to Advisor Central > Memory Advisors. The Memory Advisors page is displayed in the slide.

After Automatic Memory Management is enabled, you can see the graphical representation of the history of your memory size components in the Allocation History section of the Memory Advisors page. The top portion in the first histogram is tunable PGA only and the lower portion is all of SGA. The top portion in the second histogram is the shared pool size and the lower portion corresponds to the buffer cache.

On this page, you can also access the memory target advisor by clicking the Advice button. This advisor gives you the possible DB time improvement for various total memory sizes.

Note: You can also look at the memory target advisor by using the V\$MEMORY_TARGET_ADVISOR view.

Monitoring Automatic Memory Management

If you want to monitor the decisions made by Automatic Memory Management via a command line:

- `V$MEMORY_DYNAMIC_COMPONENTS` has the current status of all memory components
- `V$MEMORY_RESIZE_OPS` has a circular history buffer of the last 800 memory resize requests
- `V$MEMORY_TARGET_ADVICE` provides tuning advice for the `MEMORY_TARGET` initialization parameter

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring Automatic Memory Management (continued)

The dynamic performance view `V$MEMORY_DYNAMIC_COMPONENTS` shows the current sizes of all dynamically tuned memory components, including the total sizes of the SGA and instance PGA. The `V$MEMORY_TARGET_ADVICE` view provides tuning advice for the `MEMORY_TARGET` initialization parameter.

When viewing the `V$MEMORY_TARGET_ADVICE` view, the row with the `MEMORY_SIZE_FACTOR` of 1 shows the current size of memory, as set by the `MEMORY_TARGET` initialization parameter, and the amount of DB time required to complete the current workload. In previous and subsequent rows, the results show a number of alternative `MEMORY_TARGET` sizes. For each alternative size, the database shows the size factor (the multiple of the current size), and the estimated DB time to complete the current workload if the `MEMORY_TARGET` parameter were changed to the alternative size. Notice that for a total memory size smaller than the current `MEMORY_TARGET` size, estimated DB time increases.

Efficient Memory Usage: Guidelines

- Fit the SGA into physical memory.
- Tune for a high buffer cache hit ratio, with the following caveats:
 - Even valid and necessary full table scans lower it.
 - It is possible that unnecessary repeated reads of the same blocks are artificially raising it.
- Use the Memory Advisors.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Efficient Memory Usage: Guidelines

If possible, it is best to fit the SGA into physical memory, which provides the fastest access. Even though the OS may provide additional virtual memory, that memory, by its nature, can often be swapped out to disk. On some platforms, you can use the `LOCK_SGA` initialization parameter to lock the SGA into physical memory. This parameter cannot be used in conjunction with AMM or ASMM.

When a SQL statement executes, data blocks are requested for reading or writing, or both. This is considered a logical I/O. As the block is requested, the block is checked to see whether it already exists in memory. If it is not in memory, it is read from the disk, which is called a physical I/O. The number of times the block is found already in memory compared to the total number of logical I/Os is referred to as the buffer cache hit ratio. A higher ratio is usually better because that means more blocks are being found in memory without incurring disk I/O.

It is not uncommon to have a buffer cache hit ratio above 99% but that does not always mean the system is well tuned. If there is a query that is executed more often than necessary, and it constantly requests the same blocks over and over again, the ratio is raised. If it is an inefficient or unnecessary query, then it artificially inflates the ratio. This is because it should not execute in that manner or that often in the first place.

Efficient Memory Usage: Guidelines (continued)

Also, consider the fact that large full table scans (a full reading of the entire table) can lower this ratio because the entire table may be read from the disk; the scan may not take advantage of the fact that some of the blocks may be in the buffer cache. So, if there are some necessary large full table scans in your application, your well-tuned database may always have a low database buffer cache hit ratio.

Use Enterprise Manager Memory Advisors. They can help you size the SGA on the basis of the activity in your particular database.

Memory Tuning Guidelines for the Library Cache

- Establish formatting conventions for developers so that SQL statements match in the cache.
- Use bind variables.
- Eliminate unnecessary duplicate SQL.
- Consider using `CURSOR_SHARING`.
- Use PL/SQL when possible.
- Cache sequence numbers.
- Pin objects in the library cache.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Memory Tuning Guidelines for the Library Cache

The library cache, which is a part of the shared pool, is where the Oracle database stores all the SQL, Java code, PL/SQL procedures and packages, and control structures such as locks and library cache handles. The code goes into this central location so that it can be shared among all users. The benefit of sharing is that all users can take advantage of any work that is already done on behalf of SQL. Therefore, tasks such as parsing the statement and determining the data access path (also known as the “explain plan”) are done only once per statement, no matter how many times the statement is executed, and no matter how many users execute it. A library cache that is too small does not have room for all the statements being executed and, therefore, you cannot take advantage of this sharing of work for some statements. A library cache that is too large causes a burden on the system just to manage its contents.

A library cache may end up being filled with what appear to be different statements when, in fact, they are copies of the same statement. A common cause of this is having slightly different formatting for each statement. There is no match if the string does not compare exactly. Another cause is the use of literals instead of bind variables. If the only difference between two statements is literal values, then, in most cases, each of those statement executions and the overall system would benefit from replacing those literals with bind variables.

Memory Tuning Guidelines for the Library Cache (continued)

The `CURSOR_SHARING` initialization parameter can be set to have the system automatically replace literals with bind variables when statements otherwise match. You should typically take advantage of this setting as a temporary measure until the application is corrected to use bind variables where appropriate. As with all of these guidelines, the use of this variable can have other side effects, which you should investigate.

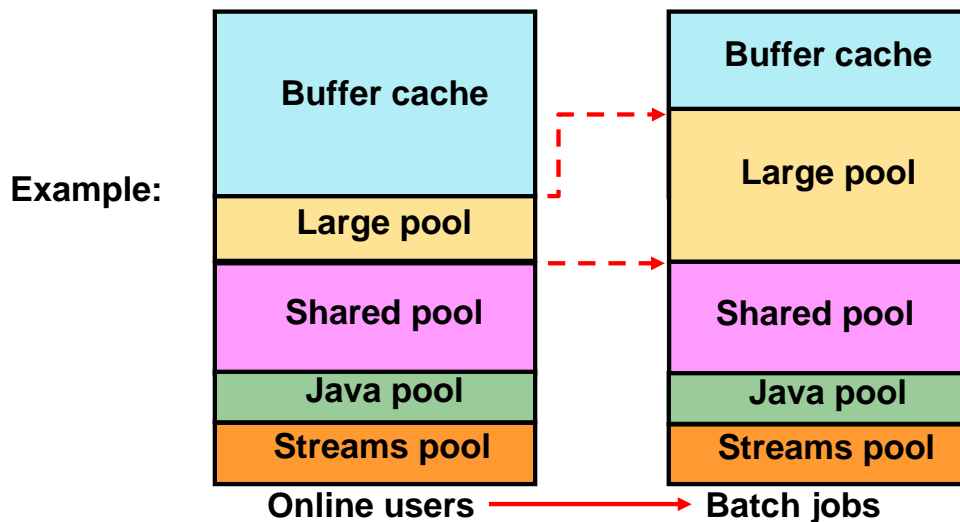
Rather than having the same SQL statement issued from several different places in an application, put the statement or statements into a stored procedure by using PL/SQL. Then just call the procedure. This guarantees that the SQL statement is shared because it exists only in one location. Also, the SQL is already parsed and has an explain plan because it is in an already compiled stored procedure.

Sequence numbers can be cached. Therefore, if there are some sequences with high activity, determine a good setting for the cache size and take advantage of it.

You can use the `DBMS_SHARED_POOL` package to pin objects in the library cache. This reduces the chance of reloading and recompiling objects. Refer to the *PL/SQL Packages and Types Reference* document for more information about how to use that package.

Automatic Shared Memory Management: Overview

- Automatically adapts to workload changes
- Maximizes memory utilization
- Helps eliminate out-of-memory errors



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Automatic Shared Memory Management: Overview

If AMM does not work for you, because you need a fixed PGA, then consider the use of Automatic Shared Memory Management (ASMM) which simplifies SGA memory management. You specify the total amount of SGA memory available to an instance using the `SGA_TARGET` initialization parameter and Oracle Database automatically distributes this memory among the various SGA components to ensure the most effective memory utilization.

For example, in a system that runs large online transactional processing (OLTP) jobs during the day (requiring a large buffer cache) and runs parallel batch jobs at night (requiring a large value for the large pool), you would have to simultaneously configure both the buffer cache and the large pool to accommodate your peak requirements.

With ASMM, when the OLTP job runs, the buffer cache grabs most of the memory to allow for good I/O performance. When the data analysis and reporting batch job starts up later, the memory is automatically migrated to the large pool so that it can be used by parallel query operations without producing memory overflow errors.

The Oracle Database remembers the sizes of the automatically tuned components across instance shutdowns if you are using a server parameter file (SPFILE). As a result, the system does not need to learn the characteristics of the workload again each time an instance is started. It can begin with information from the past instance and continue evaluating workload where it left off at the last shutdown.

How ASMM Works

- ASMM is based on workload information that MMON captures in the background.
- MMON uses memory advisors.
- Memory is moved to where it is needed the most by MMAN.
- If an SPFILE is used (which is recommended):
 - Component sizes are saved across shutdowns
 - Saved values are used to bootstrap component sizes
 - There is no need to relearn optimal values

ORACLE

Copyright © 2009, Oracle. All rights reserved.

How ASMM Works

The Automatic Shared Memory Management feature uses the SGA memory broker that is implemented by two background processes: Manageability Monitor (MMON) and Memory Manager (MMAN). Statistics and memory advisory data are periodically captured in memory by MMON. MMAN coordinates the sizing of the memory components according to MMON decisions. The SGA memory broker keeps track of the sizes of the components and pending resize operations.

The SGA memory broker observes the system and workload in order to determine the ideal distribution of memory. It performs this check every few minutes so that memory can always be present where needed. In the absence of Automatic Shared Memory Management, components had to be sized to anticipate their individual worst-case memory requirements.

On the basis of workload information, Automatic Shared Memory Management:

- Captures statistics periodically in the background
- Uses memory advisors
- Performs what-if analysis to determine the best distribution of the memory
- Moves memory to where it is most needed
- Saves component sizes across shutdown if an SPFILE is used (the sizes can be resurrected from before the last shutdown)

Enabling Automatic Shared Memory Management

To enable ASMM from manual shared memory management:

1. Get a value for **SGA_TARGET**:

```
SELECT ((SELECT SUM(value) FROM V$SGA) - (SELECT CURRENT_SIZE  
FROM V$SGA_DYNAMIC_FREE_MEMORY)) "SGA_TARGET" FROM DUAL;
```

2. Use that value to set **SGA_TARGET**.
3. Set the values of the automatically sized SGA components to 0.

To switch to ASMM from Automatic Memory Management:

1. Set the **MEMORY_TARGET** initialization parameter to 0.
2. Set the values of the automatically sized SGA components to 0.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Enabling Automatic Shared Memory Management

The procedure for enabling ASMM differs depending on whether you are changing to ASMM from manual shared memory management or from automatic memory management. To change to ASMM from manual shared memory management:

1. Run the following query to obtain a value for **SGA_TARGET**:

```
SELECT ((SELECT SUM(value) FROM V$SGA) - (SELECT CURRENT_SIZE FROM  
V$SGA_DYNAMIC_FREE_MEMORY)) "SGA_TARGET" FROM DUAL;
```
2. Set the value of **SGA_TARGET**:

```
ALTER SYSTEM SET SGA_TARGET=value [SCOPE={SPFILE|MEMORY|BOTH}]
```

where *value* is the value computed in step 1 or is some value between the sum of all SGA component sizes and **SGA_MAX_SIZE**.
3. Set the values of the automatically sized SGA components to 0. Do this by editing the text initialization parameter file or by issuing **ALTER SYSTEM** statements. Restart the instance if required.

To change to ASMM from automatic memory management:

1. Set the **MEMORY_TARGET** initialization parameter to 0.

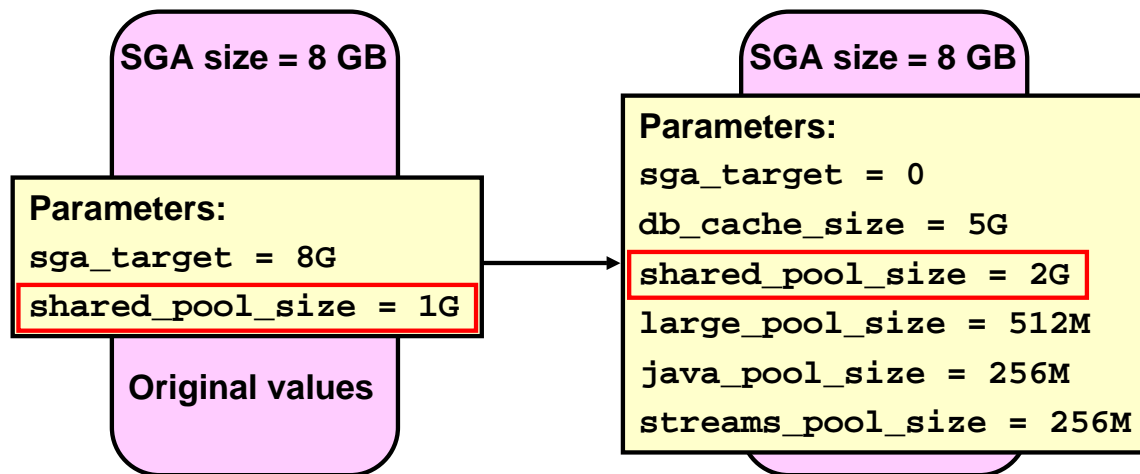
```
ALTER SYSTEM SET MEMORY_TARGET = 0;
```

The database sets **SGA_TARGET** based on current SGA memory allocation.
2. Set the values of the automatically sized SGA components to 0. Restart the instance when finished.

Note: Automatic Memory Management is discussed later in this lesson.

Disabling ASMM

- Setting `SGA_TARGET` to 0 disables autotuning.
- Autotuned parameters are set to their current sizes.
- The SGA size as a whole is unaffected.



ORACLE

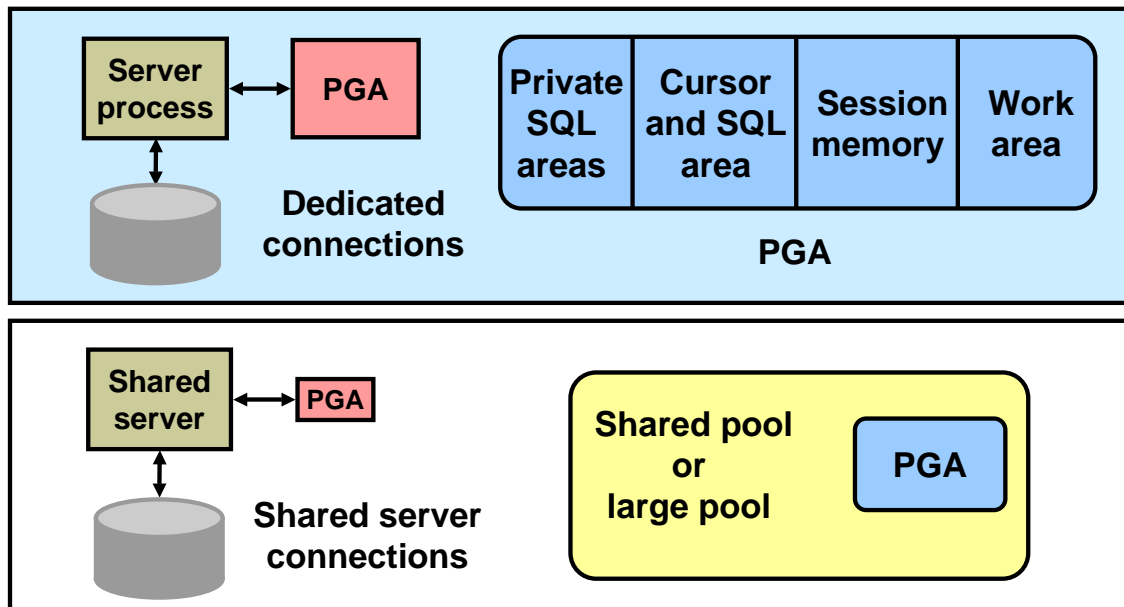
Copyright © 2009, Oracle. All rights reserved.

Disabling ASMM

You can dynamically choose to disable Automatic Shared Memory Management by setting `SGA_TARGET` to 0. In this case, the values of all the autotuned parameters are set to the current sizes of the corresponding components, even if the user had earlier specified a different nonzero value for an autotuned parameter.

In the example in the slide, the value of `SGA_TARGET` is 8 GB and the value of `SHARED_POOL_SIZE` is 1 GB. If the system has internally adjusted the size of the shared pool component to 2 GB, then setting `SGA_TARGET` to 0 results in `SHARED_POOL_SIZE` being set to 2 GB, thereby overriding the original user-specified value.

Program Global Area (PGA)



Automatic PGA memory management is enabled by default.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Program Global Area (PGA)

The Program Global Area (PGA) is a memory region that contains data and control information for a server process. It is nonshared memory created by the Oracle server when a server process is started. Access to it is exclusive to that server process. The total PGA memory allocated by all server processes attached to an Oracle instance is also referred to as the *aggregated PGA* memory allocated by the instance.

Part of the PGA can be located in the SGA when using shared servers.

PGA memory typically contains the following:

Private SQL Area

A private SQL area contains data such as bind information and run-time memory structures. This information is specific to each session's invocation of the SQL statement; bind variables hold different values, and the state of the cursor is different, among other things. Each session that issues a SQL statement has a private SQL area. Each user that submits the same SQL statement has his or her own private SQL area that uses a single shared SQL area. Thus, many private SQL areas can be associated with the same shared SQL area. The location of a private SQL area depends on the type of connection established for a session. If a session is connected through a dedicated server, private SQL areas are located in the server process's PGA. However, if a session is connected through a shared server, part of the private SQL area is kept in the SGA.

Program Global Area (PGA) (continued)

Cursor and SQL Areas

The application developer of an Oracle Pro*C program or Oracle Call Interface (OCI) program can explicitly open *cursors* or handles to specific private SQL areas, and use them as a named resource throughout the execution of the program. Recursive cursors that the database issues implicitly for some SQL statements also use shared SQL areas.

Work Area

For complex queries (for example, decision support queries), a big portion of the PGA is dedicated to work areas allocated by memory-intensive operators, such as:

- Sort-based operators, such as ORDER BY, GROUP BY, ROLLUP, and window functions
- Hash-join
- Bitmap merge
- Bitmap create
- Write buffers used by bulk load operations

A sort operator uses a work area (the sort area) to perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (the hash area) to build a hash table from its left input.

The size of a work area can be controlled and tuned. Generally, bigger work areas can significantly improve the performance of a particular operator at the cost of higher memory consumption.

Session Memory

Session memory is the memory allocated to hold a session's variables (logon information) and other information related to the session. For a shared server, the session memory is shared and not private.

Automatic PGA Memory Management

By default, Oracle Database automatically and globally manages the total amount of memory dedicated to the instance PGA. You can control this amount by setting the initialization parameter `PGA_AGGREGATE_TARGET`. Oracle Database then tries to ensure that the total amount of PGA memory allocated across all database server processes and background processes never exceeds this target.

Using the V\$PARAMETER View

```
SGA_TARGET = 8G
```

```
DB_CACHE_SIZE = 0  
JAVA_POOL_SIZE = 0  
LARGE_POOL_SIZE = 0  
SHARED_POOL_SIZE = 0  
STREAMS_POOL_SIZE = 0
```

```
SELECT name, value, isdefault  
FROM   v$parameter  
WHERE  name LIKE '%size';
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the V\$PARAMETER View

When you specify a nonzero value for SGA_TARGET and do not specify a value for an autotuned SGA parameter, the value of the autotuned SGA parameters in the V\$PARAMETER view is 0, and the value of the ISDEFAULT column is TRUE.

If you have specified a value for any of the autotuned SGA parameters, the value that is displayed when you query V\$PARAMETER is the value that you specified for the parameter.

Quiz

For best performance, you should enable both Automatic Memory Management (AMM) and Automatic Shared Memory Management (ASMM) by setting the `MEMORY_TARGET` and the `SGA_TARGET` parameters.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Summary

In this lesson, you should have learned how to:

- Describe the memory components in the SGA
- Implement Automatic Memory Management
- Manually configure SGA parameters
- Use automatic PGA memory management

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 13 Overview: Using AMM to Correct a Memory Allocation Problem

This practice covers the following topics:

- Diagnosing a memory allocation problem
- Enabling and implementing Automatic Memory Management

ORACLE

Copyright © 2009, Oracle. All rights reserved.

14

Managing Database Performance

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Monitor the performance of sessions and services
- Describe the benefits of Database Replay

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Tuning Activities

The three activities in performance management are:

- Performance planning
- Instance tuning
- SQL tuning



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Tuning Activities

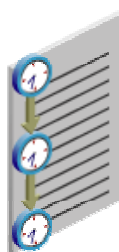
The three facets of tuning involve performance planning, instance tuning, and SQL tuning.

- Performance planning is the process of establishing the environment: the hardware, software, operating system, network infrastructure, and so on.
- Instance tuning is the actual adjustment of Oracle database parameters and operating system (OS) parameters to gain better performance of the Oracle database.
- SQL tuning involves making your application submit efficient SQL statements. SQL tuning is performed for the application as a whole, as well as for individual statements. At the application level, you want to be sure that different parts of the application are taking advantage of each other's work and are not competing for resources unnecessarily. In this lesson, you learn about some common actions that you can take to tune specific SQL statements.

Note: For more information about performance tuning, refer to the *Oracle Database Performance Tuning Guide*.

Performance Planning

- Investment options
- System architecture
- Scalability
- Application design principles
- Workload testing, modeling, and implementation
- Deploying new applications



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performance Planning

There are many facets to performance planning. Planning must include a balance between performance (speed), cost, and reliability. You must consider the investment in your system architecture: the hardware and software infrastructure needed to meet your requirements. This, of course, requires analysis to determine the value for your given environment, application, and performance requirements. For example, the number of hard drives and controllers has an impact on the speed of data access.

The ability of an application to scale is also important. This means that you are able to handle more and more users, clients, sessions, or transactions, without incurring a huge impact on overall system performance. The most obvious violator of scalability is serializing operations among users. If all users go through a single path one at a time, then as more users are added, there are definitely adverse effects on performance. This is because more and more users line up to go through that path. Poorly written SQL also affects scalability. It requires many users to wait for inefficient SQL to complete; each user competing with the other on a large number of resources that they are not actually in need of.

The principles of application design can greatly affect performance. Simplicity of design, use of views and indexes, and data modeling are all very important.

Performance Planning (continued)

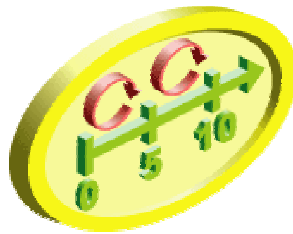
Any application must be tested under a representative production workload. This requires estimating database size and workload, and generating test data and system load.

Performance must be considered as new applications (or new versions of applications) are deployed. Sometimes design decisions are made to maintain compatibility with old systems during the rollout. A new database should be configured (on the basis of the production environment) specifically for the applications that it hosts.

A difficult and necessary task is testing the existing applications when changing the infrastructure. For example, upgrading the database to a newer version, or changing the operating system or server hardware. Before the application is deployed for production in the new configuration, you want to know the impact. The application will almost certainly require additional tuning. You need to know that the critical functionality will perform, without errors.

Instance Tuning

- Have well-defined goals.
- Allocate memory to database structures.
- Consider I/O requirements in each part of the database.
- Tune the operating system for optimal performance of the database.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Instance Tuning

At the start of any tuning activity, it is necessary to have specific goals. A goal such as “Process 500 sales transactions per minute” is easier to work toward than one that says, “Make it go as fast as you can, and we’ll know when it’s good enough.”

You must allocate Oracle database memory suitably for your application to attain optimum performance. You have a finite amount of memory to work with. Too little memory allotted to certain parts of the Oracle database can cause inefficient background activity, which you may not even be aware of without doing some analysis.

Disk I/O is often the bottleneck of a database and, therefore, requires a lot of attention at the outset of any database implementation.

The operating system configuration can also affect the performance of an Oracle database. For more information, see the *Oracle Database Installation Guide* for your particular platform.

Performance Tuning Methodology

The tuning steps:

- Tune from the top down. Tune:
 1. The design
 2. The application code
 3. The instance
- Tune the area with the greatest potential benefit. Identify and tune:
 - SQL using the greatest resources
 - The longest waits
 - The largest service times
- Stop tuning when the goal is met.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performance Tuning Methodology

Oracle has developed a tuning methodology based on years of experience. The basic steps are:

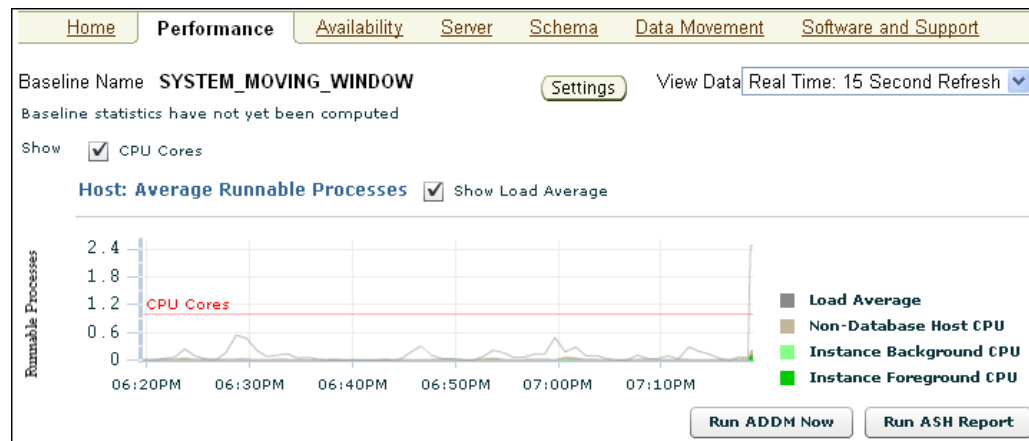
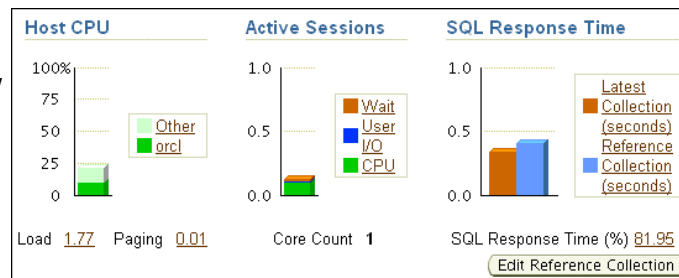
1. Check the OS statistics and general machine health before tuning the instance to be sure that the problem is in the database. Use the Enterprise Manager database home page.
2. Tune from the top down. Start with the design, then the application, and then the instance. For example, try to eliminate the full tables scans causing the I/O contention before tuning the tablespace layout on disk. This activity often requires access to the application code.
3. Tune the area with the greatest potential benefit. The tuning methodology presented in this course is simple. Identify the biggest bottleneck and tune it. Repeat this step. All the various tuning tools have some way to identify the SQL statements, resource contention, or services that are taking the most time. The Oracle database provides a time model and metrics to automate the process of identifying bottlenecks. The Advisors available in Oracle Database 11g use precisely this methodology.
4. Stop tuning when you meet your goal. This step implies that you set tuning goals.

This is a general approach to tuning the database instance and may require multiple passes.

Performance Monitoring

With Enterprise Manager:

- Performance overview
- Graphs of metrics and details



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performance Monitoring

You can respond to changes in performance only if you know the performance has changed. Oracle Database 11g provides several ways to monitor the current performance of the database instance. The database home page of Enterprise Manager (EM) provides a quick check of the health of the instance and the server, with graphs showing CPU usage, active sessions, and SQL response time. The home page also shows any alerts that have been triggered.

The Performance tab in EM shows several graphs of performance metrics from several directions. You can see performance in terms of CPU, average active sessions, throughput, I/O, and other dimensions. From the Performance page, you can follow links to detailed information: including sessions and individual SQL statements.

The information displayed in EM is based on performance views that exist in the database. You can access these views directly with SQL*Plus. Occasionally, you may need to access these views for some detail about the raw statistics.

Performance Tuning Data

Type of data gathered:

- Cumulative statistics:
 - Wait events with time information
 - Time model
- Metrics: Statistic rates
- Sampled statistics: Active session history
 - Statistics by session
 - Statistics by SQL
 - Statistics by service
 - Other dimensions



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Performance Tuning Data

The Oracle database server software captures information about its own operation. Three major types of data are collected: cumulative statistics, metrics, and sampled statistics.

Cumulative statistics are counts and timing information of a variety of events that occur in the database server. Some are quite important, such as buffer busy waits. Others have little impact on tuning, such as index block split. The most important events for tuning are usually the ones showing the greatest cumulative time values. The statistics in Oracle Database 11g are correlated by the use of a time model. The time model statistics are based on a percentage of DB time, giving them a common basis for comparison.

Metrics are statistic counts per unit. The unit could be time (such as seconds), transaction, or session. Metrics provide a base to proactively monitor performance. You can set thresholds on a metric causing an alert to be generated. For example, you can set thresholds for when the reads per millisecond exceed a previously recorded peak value or when the archive log area is 95% full.

Sampled statistics are gathered automatically when `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`. Sampled statistics allow you to look back in time. You can view session and system statistics that were gathered in the past, in various dimensions, even if you had not thought of specifying data collection for these beforehand.

Optimizer Statistics Collection

- SQL performance tuning: Depends on collection of accurate statistics
- Optimizer statistics:
 - Object statistics
 - Operating system statistics
- Ways to collect statistics:
 - Automatically: Automatic Maintenance Tasks
 - Manually: DBMS_STATS package
 - By setting database initialization parameters
 - By importing statistics from another database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Optimizer Statistics Collection

Optimizer statistics are collections of data that are specific details about database objects. These statistics are essential for the query optimizer to choose the best execution plan for each SQL statement. These statistics are gathered periodically and do not change between gatherings.

The recommended approach to gathering optimizer statistics is to allow the Oracle database to automatically gather the statistics. The Automatic Maintenance Tasks can be created automatically at database creation time and is managed by the Scheduler. It gathers statistics on all objects in the database that have either missing or stale optimizer statistics by default. You can change the default configuration through the Automatic Maintenance Tasks page.

System statistics describe the system's hardware characteristics, such as I/O and CPU performance and utilization, to the query optimizer. When choosing an execution plan, the optimizer estimates the I/O and CPU resources required for each query. System statistics enable the query optimizer to more accurately estimate I/O and CPU costs, and thereby choose a better execution plan. System statistics are collected using the DBMS_STATS.GATHER_SYSTEM_STATS procedure. When the Oracle database gathers system statistics, it analyzes system activity in a specified period of time. System statistics are not automatically gathered. Oracle Corporation recommends that you use the DBMS_STATS package to gather system statistics.

Optimizer Statistics Collection (continued)

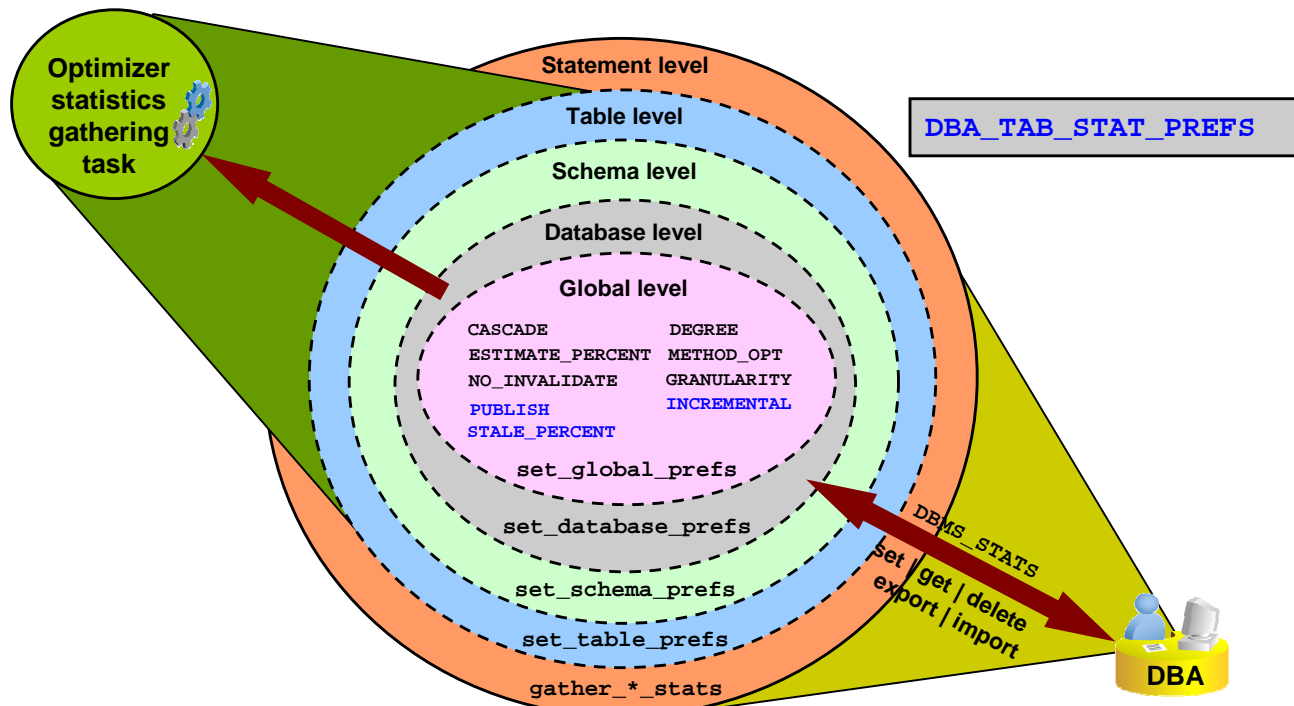
If you choose not to use automatic statistics gathering, then you must manually collect statistics in all schemas, including system schemas. If the data in your database changes regularly, you also need to gather statistics regularly to ensure that the statistics accurately represent characteristics of your database objects. To manually collect statistics, use the DBMS_STATS package. This PL/SQL package is also used to modify, view, export, import, and delete statistics.

You can also manage optimizer and system statistics collection through database initialization parameters. For example:

- The OPTIMIZER_DYNAMIC_SAMPLING parameter controls the level of dynamic sampling performed by the optimizer. You can use dynamic sampling to estimate statistics for tables and relevant indexes when they are not available or are too out of date to trust. Dynamic sampling also estimates single-table predicate selectivity when collected statistics cannot be used or are likely to lead to significant errors in estimation.
- The STATISTICS_LEVEL parameter controls all major statistics collections or advisories in the database and sets the statistics collection level for the database. The values for this parameter are BASIC, TYPICAL, and ALL. You can query the V\$STATISTICS_LEVEL view to determine which parameters are affected by the STATISTICS_LEVEL parameter.

Note: Setting STATISTICS_LEVEL to BASIC disables many automatic features and is not recommended.

Statistic Preferences: Overview



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Statistic Preferences: Overview

The automated statistics-gathering feature was introduced in Oracle Database 10g, Release 1 to reduce the burden of maintaining optimizer statistics. However, there were cases where you had to disable it and run your own scripts instead. One reason was the lack of object-level control. Whenever you found a small subset of objects for which the default gather statistics options did not work well, you had to lock the statistics and analyze them separately by using your own options. For example, the feature that automatically tries to determine adequate sample size (`ESTIMATE_PERCENT=AUTO_SAMPLE_SIZE`) does not work well against columns that contain data with very high frequency skews. The only way to get around this issue was to manually specify the sample size in your own script.

Note: You can describe all the effective statistics preference settings for all relevant tables by using the `DBA_TAB_STAT_PREFS` view.

Using Statistic Preferences

- **PUBLISH:** Used to decide whether to publish the statistics to the dictionary or to store them in a pending area before
- **STALE_PERCENT:** Used to determine the threshold level at which an object is considered to have stale statistics. The value is a percentage of rows modified since the last statistics gathering.
- **INCREMENTAL:** Used to gather global statistics on partitioned tables in an incremental way

```
exec dbms_stats.set_table_prefs('SH','SALES','STALE_PERCENT','13');
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Statistic Preferences

The Statistic Preferences feature in Oracle Database 11g introduces flexibility so that you can rely more on the automated statistics-gathering feature to maintain the optimizer statistics when some objects require settings that are different from the database default.

This feature allows you to associate the statistics-gathering options that override the default behavior of the GATHER_*_STATS procedures and the automated Optimizer Statistics Gathering task at the object or schema level. You can use the DBMS_STATS package to manage the gathering statistics options.

You can set, get, delete, export, and import those preferences at the table, schema, database, and global levels. Global preferences are used for tables that do not have preferences, whereas database preferences are used to set preferences on all tables.

The following options are new in Oracle Database 11g, Release 1:

- **PUBLISH** is used to decide whether to publish the statistics to the dictionary or to store them in a pending area before.
- **STALE_PERCENT** is used to determine the threshold level at which an object is considered to have stale statistics. The value is a percentage of rows modified since the last statistics gathering. The example changes the 10 percent default to 13 percent for SH.SALES only.
- **INCREMENTAL** is used to gather global statistics on partitioned tables in an incremental way.

Setting Global Preferences with Enterprise Manager

The screenshot displays the Oracle Enterprise Manager 11g Database Control interface. On the left, the 'Database Instance: orcl' page is shown with tabs for Home, Performance, Availability, and Server. The 'Server' tab is selected. Below this, the 'Query Optimizer' section contains links for 'Manage Optimizer Statistics' and 'SQL Plan Control'. The 'Manage Optimizer Statistics' link is highlighted with a red box. Below that, the 'Manage Optimizer Statistics' page is shown with a list of operations and related links. The 'Global Statistics Gathering Options' link is highlighted with a red box. On the right, the 'Global Statistics Gathering Options' page is shown. It includes a 'Statistics History' section with a 'Retention Period (days)' of 31. Below this is the 'Gather Optimizer Statistics Default Options' section, which contains various settings for optimizer statistics gathering, such as 'Estimate Percentage', 'Degree of Parallelism', 'Granularity', 'Cursor Invalidation', 'Cascade', 'Target Object Class (Auto Job)', 'Stale Percentage', 'Incremental', 'Publish', and 'Histograms'. The 'Apply' button is visible at the bottom right of the page.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Setting Global Preferences with Enterprise Manager

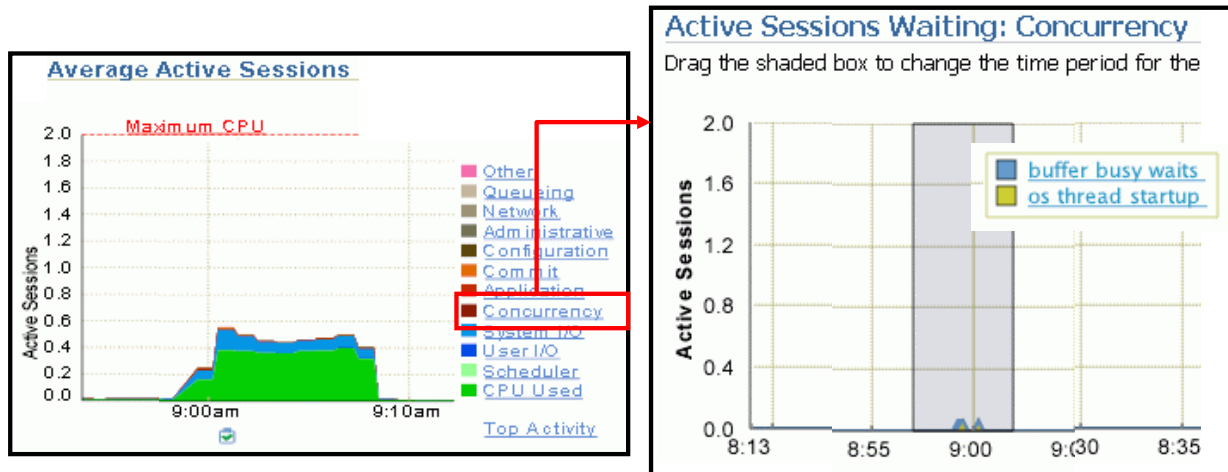
It is possible to control global preference settings by using Enterprise Manager. You do so on the Manage Optimizer Statistics page, which you access from the Database home page by clicking the Server tab, then the Manage Optimizer Statistics link, and then the Global Statistics Gathering Options link.

On the Global Statistics Gathering Options page, change the global preferences in the Gather Optimizer Statistics Default Options section. When finished, click the Apply button.

Note: To change the statistics gathering options at the object level or schema level, click the Object Level Statistics Gathering Preferences link on the Manage Optimizer Statistics page.

Oracle Wait Events

- A collection of wait events provides information about the sessions or processes that had to wait or must wait for different reasons.
- These events are listed in the `V$EVENT_NAME` view.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Oracle Wait Events

Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention. Remember that these are only symptoms of problems, not the actual causes.

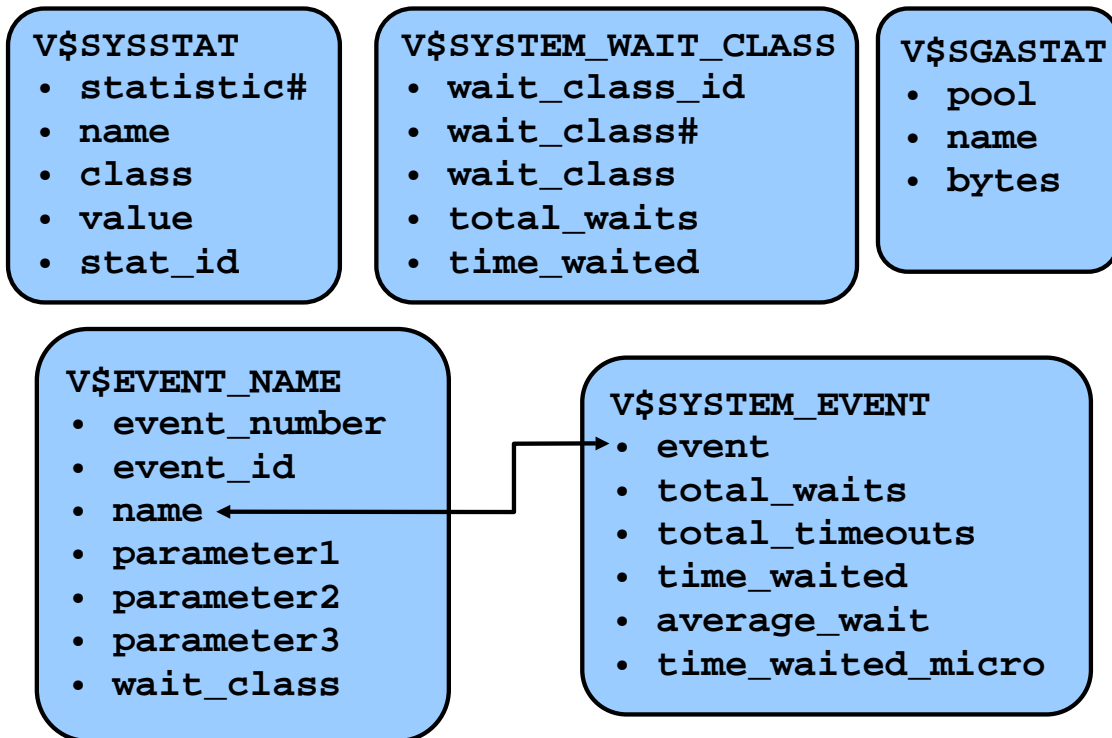
Wait events are grouped into classes. The wait event classes include: Administrative, Application, Cluster, Commit, Concurrency, Configuration, Idle, Network, Other, Scheduler, System I/O, and User I/O.

There are more than 800 wait events in the Oracle database, including free buffer wait, latch free, buffer busy waits, db file sequential read, and db file scattered read.

Using EM, you can view wait events by opening the Performance page and viewing the “Average Active Sessions” graph, as shown in the slide. By clicking the link for a particular wait event class, you can drill down to the specific wait events by using the Top Activity interface. In this example, there was a very small set of buffer busy waits.

For a list of the most common Oracle events, refer to the *Oracle Database Reference 11g* documentation.

Instance Statistics



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Instance Statistics

To effectively diagnose performance problems, statistics must be available. The Oracle database instance generates many types of cumulative statistics for the system, sessions, and individual SQL statements at the instance level. The Oracle database also tracks cumulative statistics on segments and services. When analyzing a performance problem in any of these scopes, you typically look at the change in statistics (delta value) over the period of time you are interested in.

Note: Instance statistics are dynamic and are reset at every instance startup. These statistics can be captured at a point in time and held in the database in the form of snapshots.

Wait Events Statistics

All the possible wait events are cataloged in the **V\$EVENT_NAME** view.

Cumulative statistics for all sessions are stored in **V\$SYSTEM_EVENT**, which shows the total waits for a particular event since instance startup.

When you are troubleshooting, you need to know whether a process has waited for any resource.

Systemwide Statistics

All the systemwide statistics are cataloged in the **V\$STATNAME** view: Over 400 statistics are available in Oracle Database 11g.

The server displays all calculated system statistics in the **V\$SYSSTAT** view. You can query this view to find cumulative totals since the instance started.

Instance Statistics (continued)

Example

```
SQL> SELECT name, class, value FROM v$sysstat;
NAME                                CLASS      VALUE
-----
...
table scans (short tables)          64         135116
table scans (long tables)            64           250
table scans (rowid ranges)           64            0
table scans (cache partitions)       64            3
table scans (direct read)            64            0
table scan rows gotten               64       14789836
table scan blocks gotten              64        558542
...
```

Systemwide statistics are classified by the tuning topic and the debugging purpose. The classes include general instance activity, redo log buffer activity, locking, database buffer cache activity, and so on. Each of the system statistics can belong to more than one class, so you cannot do a simple join on `V$SYSSTATS.CLASS` and `V$SYSTEM_WAIT_CLASS.WAIT_CLASS#`.

You can also view all wait events for a particular wait class by querying `V$SYSTEM_WAIT_CLASS`, as in this example (with formatting applied):

```
SQL> SELECT * FROM V$SYSTEM_WAIT_CLASS
2  WHERE wait_class LIKE '%I/O%';

CLASS_ID  CLASS#  WAIT_CLASS  TOTAL_WAITS  TIME_WAITED
-----
1740759767      8  User I/O      1119152      39038
4108307767      9  System I/O    296959      27929
```

SGA Global Statistics

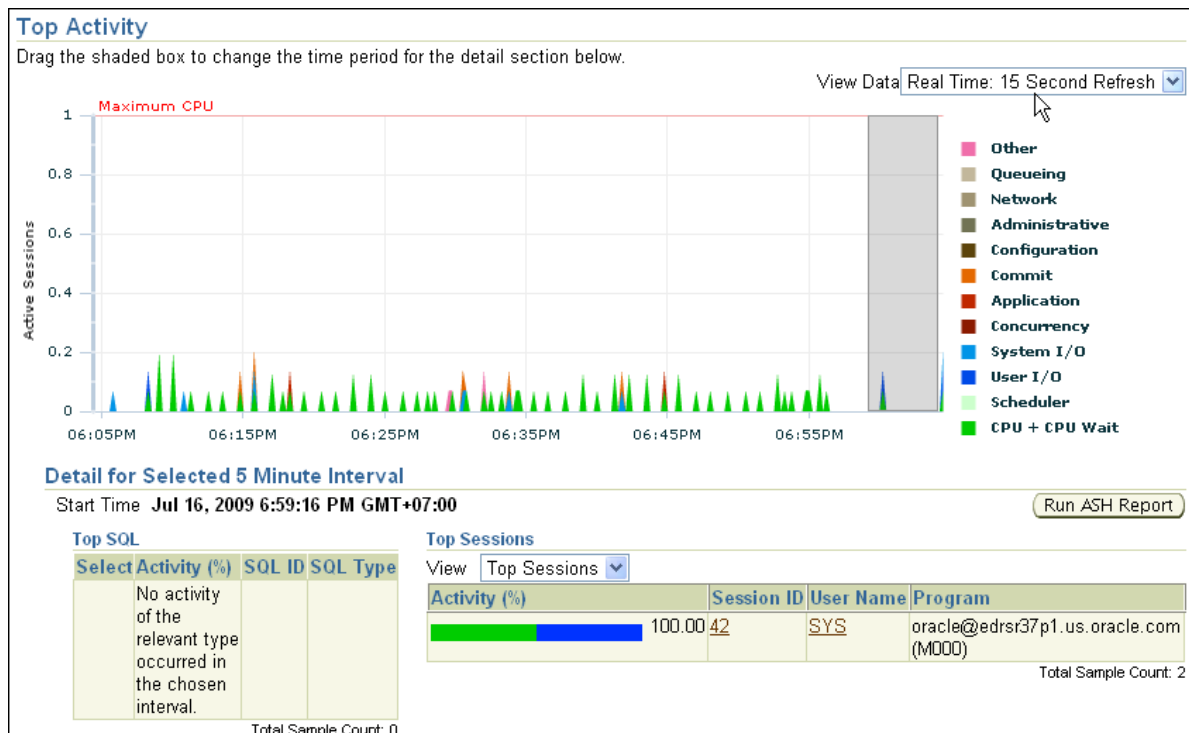
The server displays all calculated memory statistics in the `V$SGASTAT` view. You can query this view to find cumulative totals of detailed SGA usage since the instance started, as in the following example:

```
SQL> SELECT * FROM v$sgastat;
POOL      NAME                                BYTES
-----
fixed_sga                                7780360
buffer_cache                            25165824
log_buffer                               262144
shared pool sessions                    1284644
shared pool sql area                     22376876
...
```

The results shown are only a partial display of the output.

When the `STATISTICS_LEVEL` parameter is set to `BASIC`, the value of the `TIMED_STATISTICS` parameter defaults to `FALSE`. Timing information is not collected for wait events and much of the performance-monitoring capability of the database is disabled. The explicit setting of `TIMED_STATISTICS` overrides the value derived from `STATISTICS_LEVEL`.

Monitoring Session Performance



ORACLE

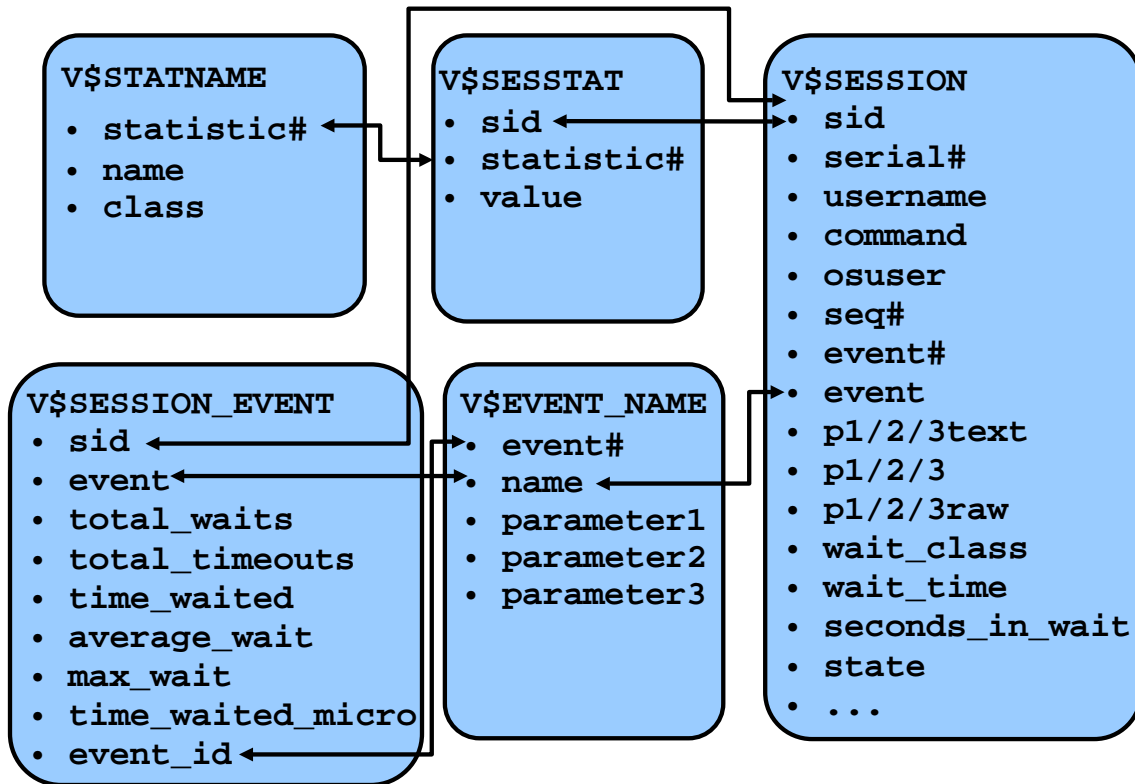
Copyright © 2009, Oracle. All rights reserved.

Monitoring Session Performance

Enterprise Manager provides session detail pages so that you can view the wait events occurring in individual sessions. On the Performance tabbed page, click Top Activity to view the summary of all sessions. In the lower right corner of the Top Activity page is a listing of the Top Sessions. Click the session identifier to view the Session Details page.

The Top Activity page and Session Details page are based on performance view in the database.

Displaying Session-Related Statistics



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Displaying Session-Related Statistics

You can display current session information for each user logged on by querying **V\$SESSION**. For example, you can use **V\$SESSION** to determine whether a session represents a user session, or was created by a database server process (**BACKGROUND**).

You can query either **V\$SESSION** or **V\$SESSION_WAIT** to determine the resources or events for which active sessions are waiting.

You can view user session statistics in **V\$SESSTAT**. The **V\$SESSION_EVENT** view lists information about waits for an event by a session.

Cumulative values for instance statistics are generally available through dynamic performance views, such as **V\$SESSTAT** and **V\$SYSSTAT**. Note that the cumulative values in dynamic views are reset when the database instance is shut down.

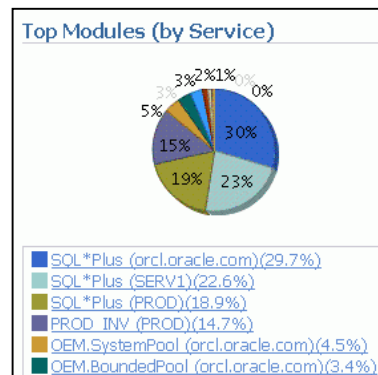
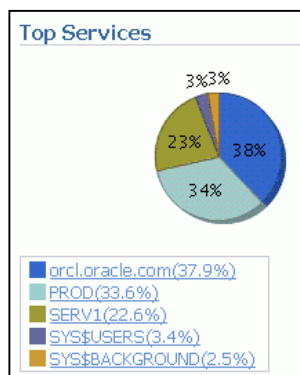
The **V\$MYSTAT** view displays the statistics of the current session.

You can also query **V\$SESSMETRIC** to display the performance metric values for all active sessions. This view lists performance metrics such as CPU usage, number of physical reads, number of hard parses, and the logical read ratio.

Displaying Service-Related Statistics

For *n*-tier environments, because session statistics are not as helpful, you can see service-level statistics in these views:

- **V\$SERVICE_EVENT**: Aggregated wait counts and wait times for each service, on a per event basis
- **V\$SERVICE_WAIT_CLASS**: Aggregated wait counts and wait times for each service on a wait class basis



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Displaying Service-Related Statistics

In an *n*-tier environment where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately. These two views provide the same information that their like-named session counterparts provide, except that the information is presented at the service level rather than at the session level.

V\$SERVICE_WAIT_CLASS shows wait statistics for each service, broken down by wait class.

V\$SERVICE_EVENT shows the same information as V\$SERVICE_WAIT_CLASS, except that it is further broken down by event ID.

Enterprise Manager also provides aggregation by service and by module and service. You can click the legend in each of the views, to view the activity and statistics for each service.

You can define a service in the database by using the DBMS_SERVICE package and use the net service name to assign applications to a service.

Troubleshooting and Tuning Views

Instance/Database

V\$DATABASE
V\$INSTANCE
V\$PARAMETER
V\$SPPARAMETER
V\$SYSTEM_PARAMETER
V\$PROCESS
V\$BGPROCESS
V\$PX_PROCESS_SYSSTAT
V\$SYSTEM_EVENT

Memory

V\$BUFFER_POOL_STATISTICS
V\$LIBRARYCACHE
V\$SGAINFO
V\$PGASTAT

Disk

V\$DATAFILE
V\$FILESTAT
V\$LOG
V\$LOG_HISTORY
V\$DBFILE
V\$TEMPFILE
V\$TEMPSEG_USAGE
V\$SEGMENT_STATISTICS

Contention

V\$LOCK
V\$UNDOSTAT
V\$WAITSTAT
V\$LATCH

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Troubleshooting and Tuning Views

The slide lists some of the views you may need to access to determine the cause of performance problems or analyze the current status of your database. Many of these views show the same data that is used in creating the reports that the Enterprise Manager produces from the Automatic Workload Repository (AWR). In some cases, the raw data is needed to fully diagnose a problem.

For a complete description of these views, refer to the *Oracle Database Reference Manual*.

Dictionary Views

- The following dictionary and special views display object statistics after use of the DBMS_STATS package:
 - DBA_TABLES , DBA_TAB_COLUMNS
 - DBA_CLUSTERS
 - DBA_INDEXES
 - DBA_TAB_HISTOGRAMS
- This statistical information is static until you reexecute the appropriate procedures in DBMS_STATS.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Dictionary Views

When you need to look at the optimizer statistics of specific database objects in detail, use the DBMS_STATS package, which collects statistics and populates columns in some DBA_xxx views.

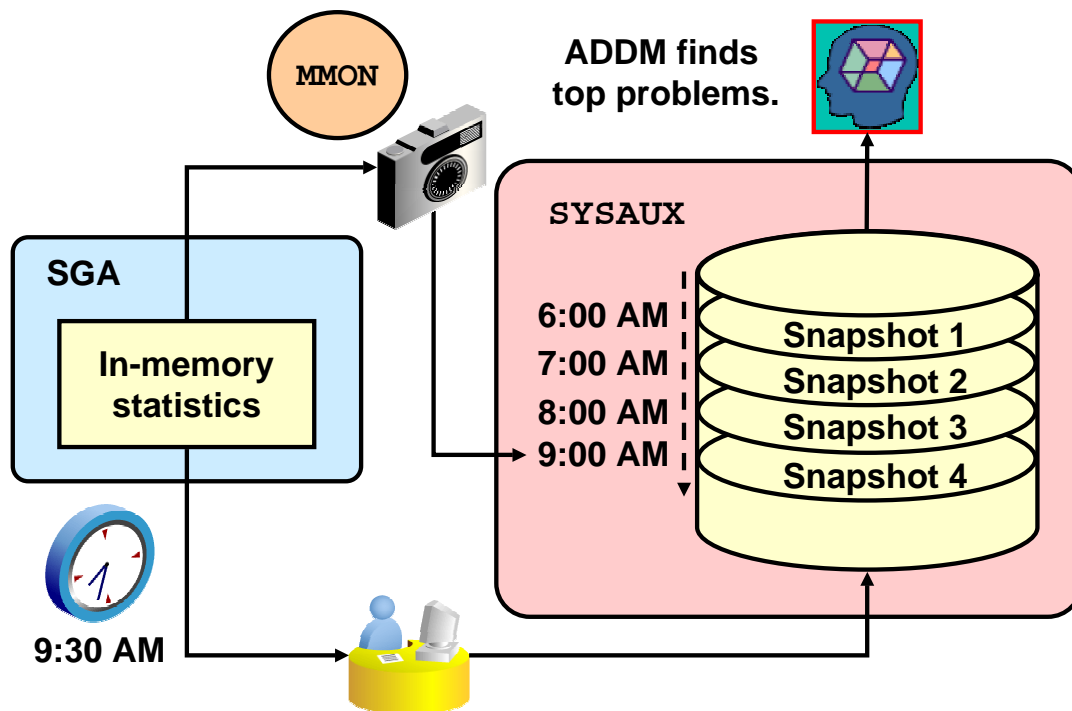
DBMS_STATS populates columns in the views concerned with:

- Table data storage within extents and blocks:
 - DBA_TABLES
 - DBA_TAB_COLUMNS
- Cluster data storage within extents and blocks:
 - DBA_CLUSTERS
- Index data storage within extents and blocks, and index usefulness:
 - DBA_INDEXES
- Nonindexed and indexed columns data distribution:
 - DBA_TAB_HISTOGRAMS

For more information about using the DBMS_STATS package, refer to the *Oracle Database Performance Tuning Guide*.

Performing an ANALYZE INDEX ... VALIDATE STRUCTURE command populates the INDEX_STATS and INDEX_HISTOGRAM views that contain statistics for indexes.

Automatic Workload Repository



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Automatic Workload Repository

The Automatic Workload Repository (AWR) is a collection of persistent system performance statistics owned by SYS. The AWR resides in the SYSAUX tablespace.

A *snapshot* is a set of performance statistics captured at a certain time and stored in the AWR. Each snapshot is identified by a snapshot sequence number (`snap_id`) that is unique in the AWR. By default, snapshots are generated every 60 minutes. You can adjust this frequency by changing the snapshot `INTERVAL` parameter. Because the database advisors rely on these snapshots, be aware that adjustment of the interval setting can affect diagnostic precision. For example, if the `INTERVAL` is set to 4 hours, you may miss transient events that would be noticeable in 60-minute intervals.

You can use the `DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS` stored procedure or Database Control to change the settings that control snapshot collection. In Database Control, click Automatic Workload Repository in the Statistics Management region of the Server tabbed page. Then click Edit to make the changes. The stored procedure offers more flexibility in defining `INTERVAL` values than does Database Control.

You can take manual snapshots by using Database Control or the `DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT` stored procedure. Taking manual snapshots is supported in conjunction with the automatic snapshots that the system generates. Manual snapshots are expected to be used when you want to capture the system behavior at two specific points in time that do not coincide with the automatic schedule.

Automatic Workload Repository (continued)

Statspack is a bundled utility that provides a subset of the collection and reporting capability of the AWR. However, there is no supported path to migrate Statspack data into the workload repository. Also, the workload repository is not compatible with the Statspack schema. Statspack is not accessible through Enterprise Manager; it requires setup, and does not have automatic retention settings, or automatic purge. The Statspack utility does provide scripts for setup, automatic snapshot collection, and reporting. Statspack snapshots can be marked for retention, as part of a Statspack baseline, or purged with provided scripts.

Statspack is documented in the `$ORACLE_HOME/rdbms/admin/spdoc.txt` file.

Using Automatic Workload Repository Views

- DBA_HIST_DB_CACHE_ADVICE
- DBA_HIST_DISPATCHER
- DBA_HIST_DYN_REMASTER_STATS
- DBA_HIST_IOSTAT_DETAIL
- DBA_HIST_SHARED_SERVER_SUMMARY

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using New Automatic Workload Repository Views

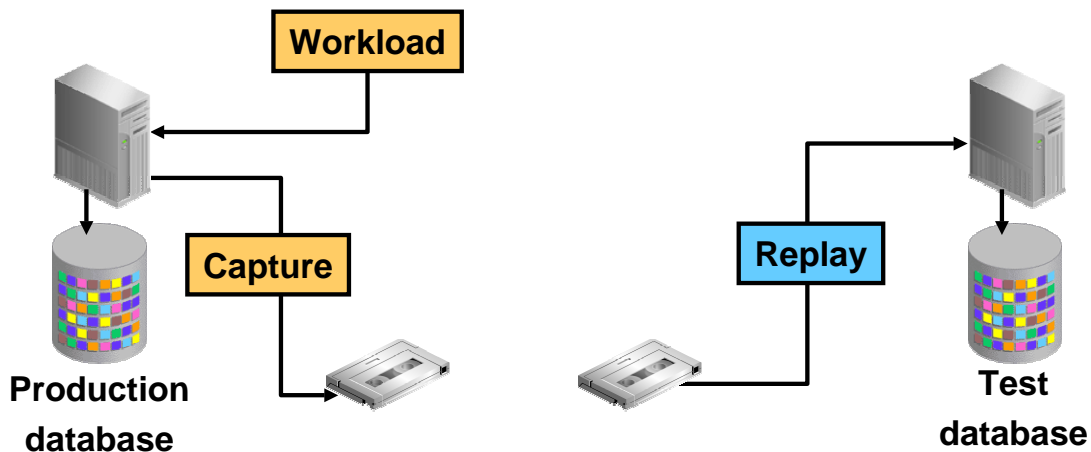
The following AWR statistics views are available in Oracle Database 11g R2:

- DBA_HIST_DB_CACHE_ADVICE: Displays historical predictions of the number of physical reads for the cache size corresponding to each row
- DBA_HIST_DISPATCHER: Displays historical information for each dispatcher process at the time of the snapshot
- DBA_HIST_DYN_REMASTER_STATS: Displays statistical information about the dynamic remastering process
- DBA_HIST_IOSTAT_DETAIL: Displays historical I/O statistics aggregated by file type and function
- DBA_HIST_SHARED_SERVER_SUMMARY: Displays historical information for shared servers, such as shared server activity, common queues, and dispatcher queues

Real Application Testing Overview: Database Replay

Database Replay:

- Captures production workloads
- Tests with realistic workloads
- Replays the same SQL against the same data in each test



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Real Application Testing Overview: Database Replay

The system is going to change. The hardware must be upgraded, or the operating system, or the database version, or all of them. You have to assure the users that the application will continue to function after the upgrade, and often guarantee performance at least as good as it was before the upgrade.

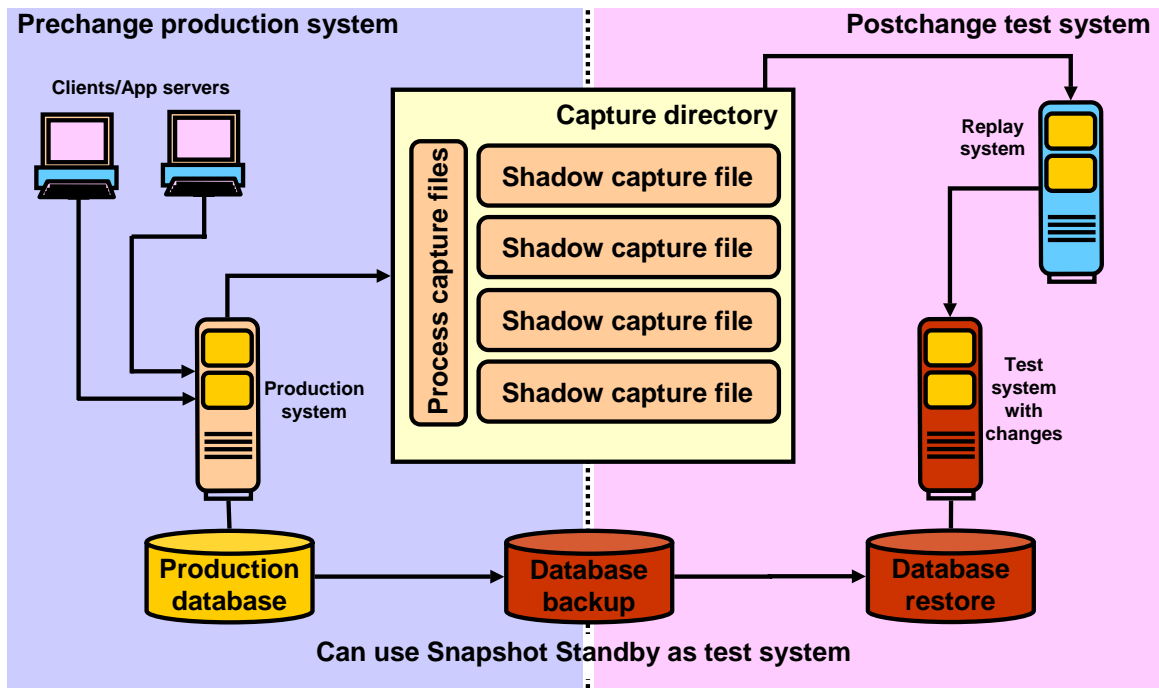
You build a test environment and test the application with a workload. You make changes and test again until you are confident that the application is working in the new environment.

This whole process depends on the test workload being representative of the actual production workload. A single statement in the production workload missing in the test workload could cause the upgrade to fail in production.

Database Replay captures the actual production workload and allows you to replay the workload in a test environment. The replay can be modified to allow for higher throughput, asynchronous application, and faster or slower replay. Database Replay can also be used to fine-tune by running exactly the same set of statements repeatedly. You can make changes to the environment and observe the differences. This capability also allows capture and replay of test cases to resolve errors.

Using actual production workloads allows you to be confident that the testing process is complete and accurate.

The Big Picture



ORACLE

Copyright © 2009, Oracle. All rights reserved.

The Big Picture

The significant benefit with Oracle Database 11g managing system changes is the added confidence to the business in the success of performing the change. The record and replay functionality offers confidence in the ease of upgrade during a database server upgrade. A useful application of Database Replay is to test the performance of a new server configuration. Suppose you are utilizing a single-instance database and want to move to a Real Application Clusters (RAC) setup. You can record the workload of an interesting period and then set up a RAC test system for replay. During replay, you can monitor the performance benefit of the new configuration by comparing the performance to the recorded system.

Database Replay can be used for debugging. You can record and replay sessions emulating an environment to make bugs more reproducible. Manageability feature testing is another benefit. Self-managing and self-healing systems need to implement this advice automatically (“autonomic computing model”). Multiple replay iterations allow testing and fine-tuning of the control strategies’ effectiveness and stability.

To learn more about this technology, attend the *Oracle Database 11g: Performance* course.

For more information, see also the *Oracle Database Performance Tuning Guide* and the *Oracle Database PL/SQL Packages and Types Reference*.

Quiz

Select the statements that are true about statistics collection:

1. You can manually collect statistics with the DBMS_STATS package.
2. You can automatically collect statistics by enabling Automatic Maintenance Tasks.
3. You can import statistics from another database.
4. You can collect statistics by setting database initialization parameters.
5. You can collect statistics by manually updating the data dictionary.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 2, 3, 4

Summary

In this lesson, you should have learned how to:

- Monitor the performance of sessions and services
- Describe the benefits of Database Replay

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 14 Overview: Monitoring Instance Performance

This practice covers the following topics:

- Monitoring Top Services and Sessions

ORACLE

Copyright © 2009, Oracle. All rights reserved.

15

Managing Performance by SQL Tuning

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use the SQL Tuning Advisor to:
 - Identify SQL statements that are using the most resources
 - Tune SQL statements that are using the most resources
- Use the SQL Access Advisor to tune a workload

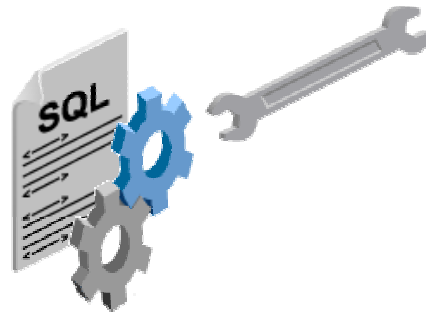
ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Tuning

SQL tuning process

- Identify poorly tuned SQL statements.
- Tune the individual statements.
- Tune the application as a whole.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Tuning

Generally, the tuning effort that yields the most benefit is SQL tuning. Poorly tuned SQL uses more resources than required. This inefficiency prevents scalability, uses more OS and database resources, and increases response time. To tune poorly tuned SQL statements, they must be identified, and then tuned. SQL statements can be tuned individually, but often the solution that optimizes one statement can hurt the performance of several others.

The SQL statements that use the most resources are by definition the statements in need of tuning. These are statements that have the longest elapsed time, use the most CPU, or do the most physical or logical reads.

Tune the individual statements by checking the optimizer statistics, check the explain plan for the most efficient access path, test alternate SQL constructions, and test possible new indexes, materialized views, and partitioning.

Test the application as a whole, using the tuned SQL statements. Is the overall performance better?

The methodology is sound, but tedious. Tuning an individual statement is not difficult. Testing the overall impact of the individual statement tuning on an application can be very difficult.

In Oracle Database 11g, a set of SQL advisors are available to identify and tune statements, individually or as a set.

SQL Advisors

Advisor Central

Advisors **Checkers**

View Data Real Time: 15 Second Refresh

Advisors

[ADDM](#) [Automatic Undo Management](#) [Data Recovery Advisor](#)
[Memory Advisors](#) [MTTR Advisor](#) [Segment Advisor](#)
[SQL Advisors](#) [SQL Performance Analyzer](#) [Streams Performance Advisor](#)

Advisor Tasks [Change Default Parameters](#)

Search
Select an advisory type and optionally enter a task name to filter the data that is displayed in your results set.

Advisory Type Task Name Advisor Runs Status

SQL Performance Analyzer Last 31 Days All Go

Re-schedule Go Previous 1-25 of 55 Next 25

	Description	User	Status	Start Time	Duration (seconds)	Expires (days)
1161_1_49	ADDM auto run:	SYS	COMPLETED	Aug 25, 2009 2:00:49 AM	1	3

ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Advisors

Oracle Database 11g provides a set of advisors for SQL: the SQL Access Advisor, the SQL Tuning Advisor, the SQL Performance Analyzer, and the SQL Repair Advisor. The AWR identifies and records statistics about the recent high-load SQL statements.

The SQL Tuning Advisor analyzes one or more SQL statements one at a time. It examines statistics, SQL profiles, indexes, materialized views, and restructured SQL. The SQL Tuning Advisor can be run manually at any time, but it is run during every maintenance window against the recent high-load SQL statements. Click Automatic SQL Tuning Results to view and implement the recommendations. This automatic job can be configured to automatically implement recommended SQL profiles for the high-load statements.

The SQL Access Advisor considers changes applied to a set of SQL statements and looks for a net gain in performance. This set can be a hypothetical set of SQL, a historical set, or a manually created set.

The SQL Performance Analyzer can be used to predict and prevent potential performance problems for any database environment change that affects the structure of the SQL execution plans.

The SQL Repair Advisor is run from the Support Workbench when a SQL statement fails with a critical error. A critical error also produces an incident. The repair advisor attempts to find and recommend a SQL patch. If no patch is found, you can continue in the Support Workbench to package the incident and submit the package to Oracle Support as a Service Request (SR).

Automatic SQL Tuning Results

Automatic SQL Tuning Result Summary

The Automatic SQL Tuning runs during system maintenance windows as an automated maintenance task, searching for ways to improve the execution plans of high-load SQL statements.

Task Status

Automatic SQL Tuning (SYS_AUTO_SQL_TUNING_TASK) is currently **Enabled** [Configure](#)
Automatic Implementation of SQL Profiles is currently **Disabled**
Highly Recommended SQL Profiles **0**

Task Activity Summary

The activity summary graph shows the benefit of the task activities on the systems high-load SQL. Only profiles that significantly improve SQL performance were implemented.

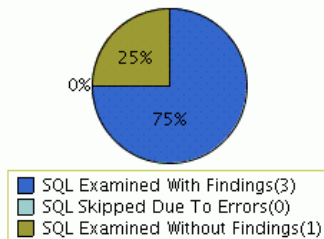
Time Period [Go](#) [View Report](#)

Begin Date **Jul 4, 2007 10:00:08 PM (UTC+07:00)** End Date **Jul 6, 2007 2:39:10 AM (UTC+07:00)**

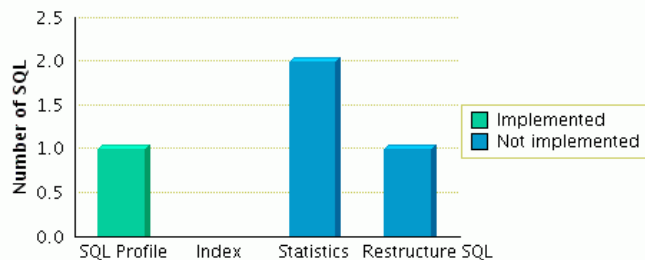
Overall Task Statistics

Executions **2** Candidate SQL **4** Distinct SQL Examined **4**

SQL Examined Status



Breakdown by Finding Type



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Automatic SQL Tuning Results

The Automatic SQL Tuning Task runs by default every night. The Automatic SQL Tuning Results link shows the result summary page. If you click **View Report**, each of the distinct SQL statements that were examined can be viewed.

Clicking the Configure button displays a page where you can change the defaults of the Automatic Tuning Task and enable automatic implementation of SQL profiles.

Implement Automatic Tuning Recommendations

Automatic SQL Tuning Result Details

Begin Date **Jul 4, 2007 10:00:08 PM (UTC +07:00)** End Date **Jul 6, 2007 2:39:10 AM (UTC +07:00)**

Recommendations

Only profiles that significantly improve SQL performance were implemented.

[View Recommendations](#) [Implement All](#)

Select	SQL Text	Parsing Schema	SQL ID	Statistics	SQL Profile	Index	Restructure SQL	Miscellaneous	Error	Date
<input checked="" type="radio"/>	SELECT NULL AS table_cat, t.owner...	SYSMAN	0prhvnva3f97z	✓	(82.9%) ✓					7/5/07

Recommendations for SQL ID:0prhvnva3f97z [Return](#)

Only one recommendation should be implemented.

SQL Text

[SELECT NULL AS table_cat, t.owner AS table_schem, t.table_name AS table_name, t.column_name AS column_name, DECODE \(t.data_type, 'CHAR', 1, 'VARCHAR2', 12, 'NUMBER', 3, ...](#)

Select Recommendation

[Original Explain Plan \(Annotated\)](#)

[Implement](#)

Select	Type	Findings	Recommendations	Rationale	Benefit (%)	New Explain Plan	Compare Explain Plans
<input checked="" type="radio"/>	SQL Profile	A potentially better execution plan was found for this statement.	Consider accepting the recommended SQL profile.		82.87	SQL	SQL

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Implement Automatic Tuning Recommendations

If you click View Report on the Automatic Tuning Results Summary page, you will see the Automatic SQL Tuning Result Details. You can implement all the recommendations or drill down to view or implement individual recommendations. On the Recommendations page, you can click the eyeglass icon on the right to see the differences that implementing a SQL profile will make in the explain plan.

SQL Tuning Advisor: Overview



SQL Tuning
Advisor

Comprehensive SQL tuning

Detect stale or missing
statistics

Tune SQL plan
(SQL profile)

Add missing index

Restructure SQL

ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Tuning Advisor: Overview

The SQL Tuning Advisor is the primary driver of the tuning process. It performs several types of analyses:

- **Statistics Analysis:** Checks each query object for missing or stale statistics, and makes recommendations to gather relevant statistics.
- **SQL Profiling:** The optimizer verifies its own estimates and collects auxiliary information to remove estimation errors. It builds a SQL profile using the auxiliary information and makes a recommendation to create it. When a SQL profile is created, it enables the query optimizer to generate a well-tuned plan.
- **Access Path Analysis:** New indexes are considered if they significantly improve access to each table in the query. When appropriate, recommendations to create such objects are made.
- **SQL Structure Analysis:** SQL statements that use bad plans are identified and relevant suggestions are made to restructure them. The suggested changes can be syntactic as well as semantic.

The SQL Tuning Advisor considers each SQL statement included in the advisor task independently. Creating a new index may help a query, but may hurt the response time of DML. So a recommended index or other object should be checked with the SQL Access Advisor over a workload (a set of SQL statements) to determine whether there is a net gain in performance.

Using the SQL Tuning Advisor

- Use the SQL Tuning Advisor to analyze SQL statements and obtain performance recommendations.
- Sources for SQL Tuning Advisor to analyze:
 - Top Activity: Analyzes the top SQL statements currently active
 - SQL Tuning Sets: Analyzes a set of SQL statements you provide
 - Historical SQL (AWR): Analyzes SQL statements from statements collected by AWR snapshots

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the SQL Tuning Advisor

The SQL Tuning Advisor runs automatically every night as the Automatic SQL Tuning Task. There may be times when a SQL statement needs immediate tuning action. You can use the SQL Tuning Advisor to analyze SQL statements and obtain performance recommendations at any time. Typically, you run this advisor as an ADDM performance-finding action.

Additionally, you can run the SQL Tuning Advisor when you want to analyze the top SQL statements consuming the most CPU time, I/O, and memory.

Even though you can submit multiple statements to be analyzed in a single task, each statement is analyzed independently. To obtain tuning recommendations that consider overall performance of a set of SQL, use the SQL Access Advisor.

SQL Tuning Advisor Options

Schedule SQL Tuning Advisor

Specify the following parameters to schedule a job to run the SQL Tuning Advisor.

* Name

Description

* SQL Tuning Set

SQL Tuning Set Description

SQL Statements Counts

Scope

Total Time Limit (minutes)

Scope of Analysis ☐ Limited
The analysis is done without SQL Profile recommendation and takes about 1 second per statement.

☒ Comprehensive
This analysis includes SQL Profile recommendation, but may take a long time.

Time Limit per Statement (minutes)

Overview

The SQL Tuning Advisor analyzes individual SQL statements, and suggests indexes, SQL profiles, restructured SQL, and statistics that improve the performance of the SQL statements.

The SQL Tuning Advisor operates on a collection of SQL. You can choose a SQL Tuning Set to run the advisor. If you do not have a SQL Tuning Set with the desired SQL for running the advisor, you can create a new one.

You can click on one of the following sources, which will lead you to a data source where you can tune SQL statements using the SQL Tuning Advisor.

[Top Activity](#)
[Historical SQL \(AWR\)](#)
[SQL Tuning Sets](#)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Tuning Advisor Options

On the Schedule SQL Tuning Advisor page, you can choose the SQL statements to include and change the automatic defaults for a tuning task. You can set the source of the SQL statements, and if you have been granted the ADVISOR system privilege, you can submit the task. Enterprise Manager then creates a tuning task for the SQL Tuning Advisor.

The SQL statement options allow you to choose one or more SQL statements from recent Top Activity, choose Historical SQL stored in the AWR, or choose from a SQL Tuning Set that you have already created.

It is important to choose the appropriate scope for the tuning task. If you choose the Limited option, then the SQL Tuning Advisor produces recommendations based on statistics check, access path analysis, and SQL structure analysis. The Limited option will not make a SQL profile recommendation. If you choose the Comprehensive option, the SQL Tuning Advisor produces all the recommendations that the Limited option produces, but it also invokes the optimizer under the SQL profiling mode to build a SQL profile. With the Comprehensive option, you can also specify a time limit for the tuning task, which by default is 30 minutes. After you select Run SQL Tuning Advisor, configure your tuning task using the SQL Tuning Options page.

SQL Tuning Advisor Recommendations

SQL Tuning Results:SQL_TUNING_1183667475959

Page Refreshed Jul 6, 2007 3:31:31 AM GMT+07:00 Refresh

Status **COMPLETED**
Started **Jul 6, 2007 3:31:23 AM**
Completed **Jul 6, 2007 3:31:28 AM**

Tuning Set Owner **SYS**
Tuning Set Name **TOP_SQL_1183667475554**
Time Limit (seconds) **1800**
Running Time (seconds) **5**

Recommendations

View

Select	SQL Text	Parsing Schema	SQL ID	Statistics	SQL Profile	Index	Restructure SQL	Miscellaneous	Error
<input checked="" type="radio"/>	DECLARE table_nonexistent EXCEPTION; PRAGMA EXCEPTION_INIT (table_nonexistent, -942); BEGIN BEG...	HR	6t4uxuxdaxpff					✓	
<input type="radio"/>	delete from sh.sales_copy	SYSTEM	0ggwcxx1quwuv	✓					
<input type="radio"/>	insert into sh.sales_copy select * from sh.sales	SYSTEM	axn4pkyybt51a	✓					

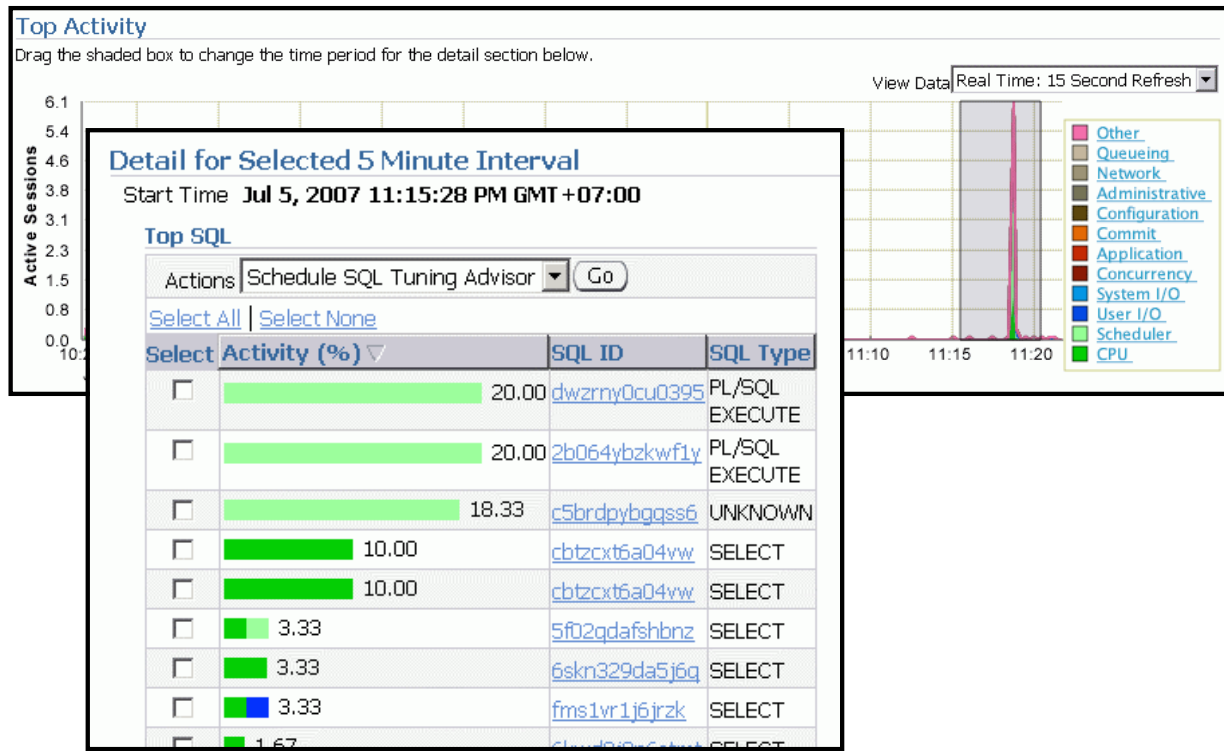
ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Tuning Advisor Recommendations

The SQL Tuning Results for the task are displayed as soon as the task completes and can also be accessed later from the Advisor Central page. A summary of the recommendations are displayed. You can review and implement individual recommendations. Select the statement and click view.

Using the SQL Tuning Advisor: Example



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the SQL Tuning Advisor: Example

You can invoke the SQL Tuning Advisor by performing the following steps:

1. Click Advisor Central in the Related Links region on the Database home page.
2. Click SQL Advisors. The SQL Tuning Advisor Links page appears.

The advisor can be run on one of the following sources:

- **Active SQL:** Analyzes the Top SQL statements currently active
- **SQL Tuning Sets:** Analyzes a set of SQL statements you provide
- **Historical SQL (AWR):** Analyzes SQL captured by snapshots in the AWR

3. Select Active SQL. Select a five-minute interval to analyze by dragging the shaded box over the target time period. Select one or more statements to analyze during the selected period.
4. Click Run SQL Tuning Advisor. The SQL Tuning Options page appears showing the SQL statements in the interval. Give your task a name and description, select Comprehensive as the scope, and select Immediately for start time. Click OK.
5. Navigate back to the Advisor Central page. The status of Advisor Tasks is listed under this heading in the Results region. Wait until your task status is completed. Check the status by clicking Refresh in your browser. Select your task and click View Result. The SQL Tuning Result page appears.
6. Select the SQL statement and click View Recommendations.

Duplicate SQL

Duplicate SQL

Page Refreshed Jul 6, 2007 4:49:59 AM GMT +07:00 [Refresh](#)

Applications can cause database to consume excessive CPU by parsing SQL statements that are similar and that can share the same SQL text. Such applications can also cause slow performance by creating contention in the database for Library Cache or Shared Pool.

CPU Consumption Since Instance Started

CPU Used as percentage of Total CPU (%) **2.19**
CPU Used for Parsing as percentage of CPU Used(%) **17.20**

Duplicate SQL Statements

This report identifies similar SQL statements that could be shared by a single SQL statement if the database application used bind variables to replace literals and SQL coding conventions to remove differences based only on character case or whitespace. You can re-write the SQL statements to gain the efficiency of using a single, shared statement.

Note: Only the first 2000 SQL statements that are executed only once are examined. The actual number of SQL statements that are duplicates can be more than 2000.

[Expand All](#) | [Collapse All](#)

Duplicates	Plan Hash Value	SQL Text
▼ Duplicates		
▼ 5	1445457117	select * from hr.employees where employee_id = 148
	1445457117	select * from hr.employees where employee_id = 148
	1445457117	select * from hr.employees where employee_id = 145
	1445457117	select * from hr.employees where employee_id = 133
	1445457117	select * from hr.employees where employee_id = 100
	1445457117	select * from hr.employees where employee_id = 132

Bind variable candidates

ORACLE

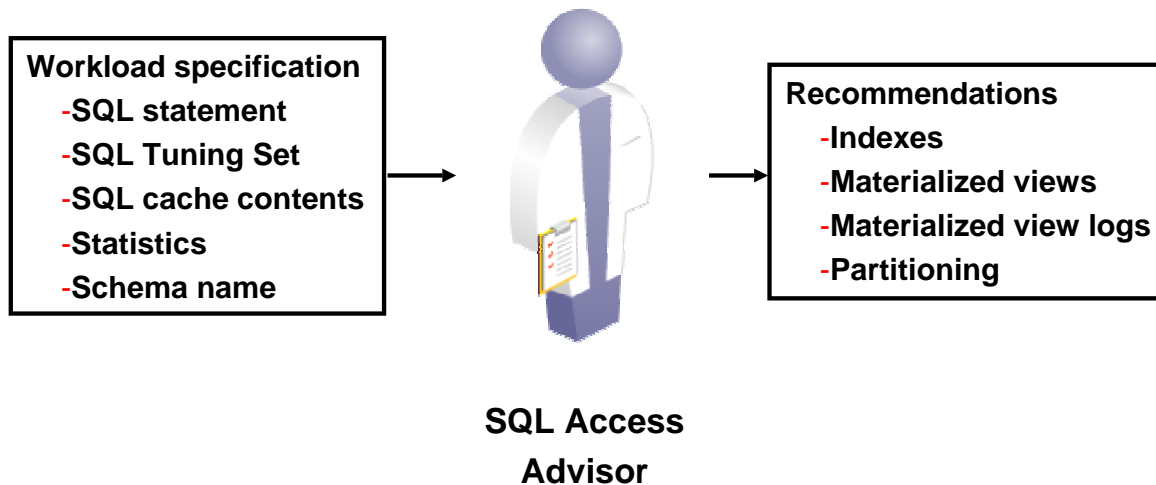
Copyright © 2009, Oracle. All rights reserved.

Duplicate SQL

Duplicate SQL statements are statements that are different only in the literal values they use or in their formatting. Statements that are different each get a separate cursor in the Library Cache. Statements that are duplicates can use the same cursor if the literals are replaced with bind variables, and the formatting is made to be the same.

Duplicate SQL statements can be identified by clicking Duplicate SQL on the Performance tabbed page in the Additional Monitoring Links section. SQL that is determined to be duplicate, except for formatting or literal differences, is listed together. This helps you determine which SQL in your application can be consolidated, thus lowering the requirements on the Library Cache and speeding up the execution of the statement.

SQL Access Advisor: Overview



ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Access Advisor: Overview

The SQL Access Advisor can recommend the proper set of materialized views, materialized view logs, partitioning, and indexes for a given workload. Understanding and using these structures is essential when optimizing SQL because they can result in significant performance improvements in data retrieval.

The SQL Access Advisor recommends bitmap, function-based, and B-tree indexes. A bitmap index offers a reduced response time for many types of ad hoc queries and reduced storage requirements compared to other indexing techniques. B-tree indexes are most commonly used in a data warehouse to index unique or near-unique keys.

Another component of the SQL Access Advisor also recommends how to optimize materialized views so that they can be fast refreshable and take advantage of general query rewrite.

Note: For more information about materialized views and query rewrite, see the *Oracle Database Performance Tuning Guide*.

Typical SQL Access Advisor Session

ORACLE Enterprise Manager 11g Database Control

Setup Preferences Help Logout

Database

Advisor Central > Logged in As SYS

SQL Access Advisor: Initial Options

Select a set of initial options.

☐ Verify use of access structures (indexes, materialized views, partitioning, etc) only

☒ Recommend new access structures

☐ Inherit Options from a previously saved Task or Template

Cancel Continue

Overview

The SQL Access Advisor evaluates SQL statements in a workload Source, and can suggest indexes, partitioning, materialized views and materialized view logs that will improve performance of the workload as a whole.

TIP You are selecting the starting point for the wizard. All options can be changed from within the wizard.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Typical SQL Access Advisor Session

When starting a SQL Access Advisor session, you can select Use Default Options and start with a predefined set of advisor options that are recommended. Additionally, you can start a task and have it inherit a set of option values as defined by a template or task by selecting “Inherit Options from a Task or Template.” These include several generic templates designed for general-purpose environments, OLTP, and data warehousing databases. You can save custom templates from a previous task and reuse them when needed.

Click Continue to launch the SQL Access Advisor Wizard.

Note: You can access SQL Access Advisor from the Advisor Central page of Database Control.

Workload Source

SQL Access Advisor: Workload Source

Database **orcl** Cancel Step 1 of 4 **Next**

Logged In As **SYS**

Select the source of the workload that you want to use for the analysis. The best workload is one that fully represents all the SQL statements that access the underlying tables.

☒ **Current and Recent SQL Activity**
SQL will be selected from the cache.

☐ **Use an existing SQL Tuning Set**
SQL Tuning Set

☐ **Create a Hypothetical Workload from the Following Schemas and Tables**
The advisor can create a hypothetical workload if the tables contain dimension or primary/foreign key constraints.
Schemas and Tables Add

Comma-separated list

☒ **TIP** Enter a schema name to specify all the tables belonging to that schema.

[Filter Options](#)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Workload Source

Use the SQL Access Advisor Wizard's Workload Source page to provide a defined workload that allows the Access Advisor to make recommendations. Supported workload sources are:

- **Current and Recent SQL Activity:** Uses the current SQL from the cache as the workload
- **Use an existing SQL Tuning Set:** Enables you to specify a previously created SQL Tuning Set as the workload source
- **Create a Hypothetical Workload from the Following Schemas and Tables:** Provides a schema that allows the advisor to search for dimension tables and produce a workload

The scope of the workload can be further reduced by applying filters that you can access in the Filter Options section. With these options, you can reduce the scope of the SQL statements that are present in the workload. The filters are applied to the workload by the advisor to focus the tuning effort.

Possible filter options are:

- Top resource consuming SQL statements
- Users, module identifier, or actions
- Tables

Recommendation Options

SQL Access Advisor: Recommendation Options

Database: **orcl** Cancel Back Step 2 of 4 **Next**

Logged In As: **sys**

Access Structures to Recommend

- ☒ Indexes
- ☒ Materialized Views
- ☐ Partitioning

Scope

The advisor can run in one of two modes, Limited or Comprehensive. Limited Mode is meant to return quickly after processing the statements with the highest cost, potentially ignoring statements with a cost below a certain threshold. Comprehensive Mode will perform an exhaustive analysis.

☐ Limited
Analysis will focus on highest cost statements

☒ Comprehensive
Analysis will be exhaustive

Advanced Options

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recommendation Options

Use the Recommendations Options page to choose whether to limit the advisor to recommendations based on a single access method. Choose Indexes, Materialized Views, Partitioning, or a combination of those from the “Access Structures to Recommend” section. You can choose Evaluation Only to evaluate only existing access structures. In this mode, the advisor does not generate new recommendations but comments on the use of existing structures. This is useful to track the effectiveness of the current index, materialized view, and MV log usage over time.

You can use the Advisor Mode section to run the advisor in one of two modes. These modes affect the quality of recommendations as well as the length of time required for processing. In Comprehensive mode, the advisor searches a large pool of candidates resulting in recommendations of the highest quality. In Limited mode, the advisor performs quickly, limiting the candidate recommendations.

Recommendation Options

▼Advanced Options

Workload Categorization

Workload Volatility

☐ Consider only queries
Choose this option if this is a Data Warehouse workload

Workload Scope

☐ Recommend dropping unused access structures
Select when workload represents all access structure use cases

Space Restrictions

Do you want to limit additional space used by recommended indexes and materialized views?

☒ No, show me all recommendations (unlimited space)

☐ Yes, limit additional space to MB Set to zero or negative to recommend dropping existing access structures to make room for better ones.

Tuning Prioritization

Prioritize tuning of SQL statements by

SQL statements will be analyzed in descending order of the value of the prioritized statistic.

☒ Consider access structures creation costs recommendations
If checked, the SQL Access Advisor will weigh the cost of creation of access structures against the frequency and potential improvement of SQL statement execution time. If unchecked, the cost of creation will be ignored. Check this box if you do not want specific recommendations generated for statements that are not executed frequently.

Default Storage Locations

By default, indexes will be placed in the schema and tablespace of the table they reference, materialized views will be placed in the schema and tablespace of the first table referenced in the query, and materialized view logs will be placed in the default tablespace of the schema of the table they reference. These fields allow you to change these default locations.

Index Tablespace

Index Schema

Materialized View Tablespace

Materialized View Schema

Materialized View Log Tablespace

Partitioning Tablespaces

Comma-separated list

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Recommendation Options (continued)

You can choose Advanced Options to show or hide options that enable you to set space restrictions, tuning options, and default storage locations. Use the Workload Categorization section to set options for Workload Volatility and Workload Scope. You can choose to favor read-only operations or you can consider the volatility of referenced objects when forming recommendations. You can also select Partial Workload, which does not include recommendations to drop unused access structures, or Complete Workload, which does include recommendations to drop unused access structures.

Use the Space Restrictions section to specify a hard space limit, which forces the advisor to produce recommendations only with total space requirements that do not exceed the specified limit.

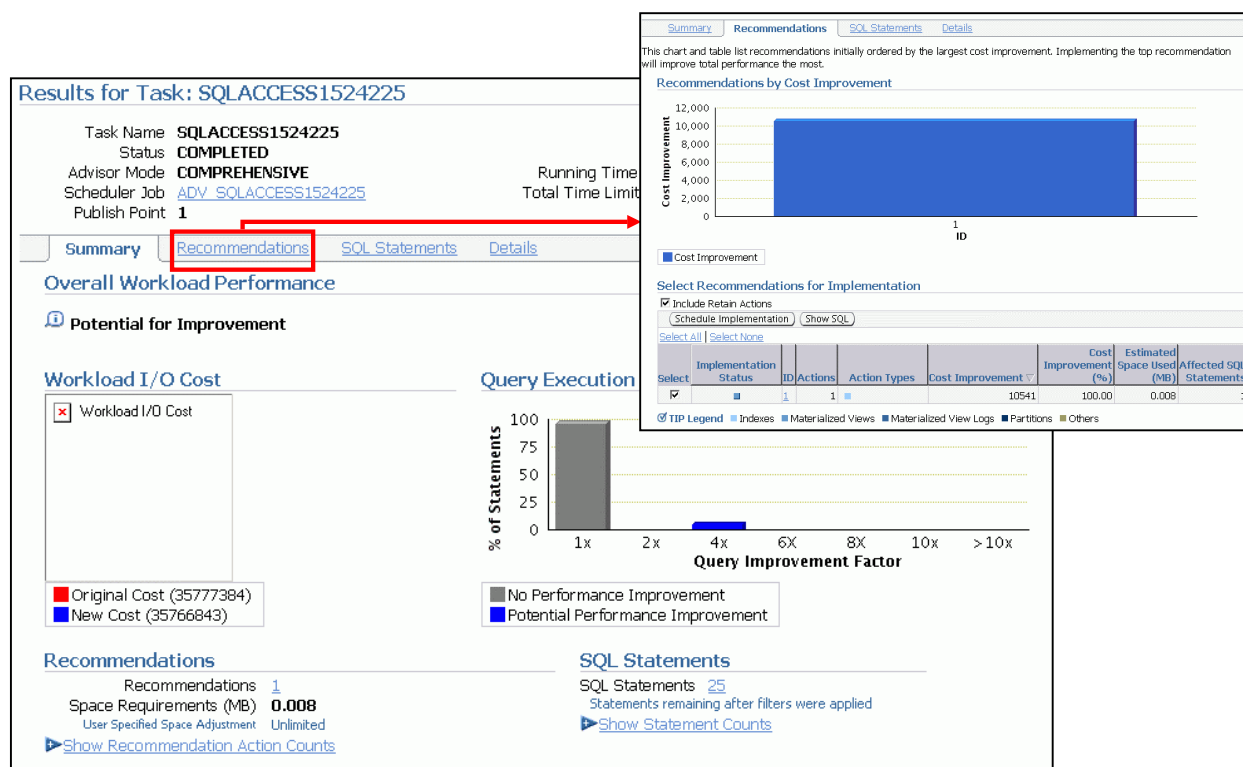
Use the Tuning Options section to specify options that customize the recommendations made by the advisor. Use the “Prioritize Tuning of SQL Statements by” drop-down list to prioritize by Optimizer Cost, Buffer Gets, CPU Time, Disk Reads, Elapsed Time, and Execution Count.

Use the Default Storage Locations section to override the defaults defined for schema and tablespace locations. By default, indexes are placed in the schema and tablespace of the table they reference. Materialized views are placed in the schema and tablespace of the user who executed one of the queries that contributed to the materialized view recommendation.

After you define these parameters, you can schedule and review your tuning task.

Oracle Database 11g: Administration Workshop II 15 - 17

Reviewing Recommendations



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Reviewing Recommendations

Using the Advisor Central page, you can list all the completed SQL Access Advisor tasks. Select the one for which you want to see the recommendations, and then click the View Result button. Use the Results for Task Summary page to get an overview of the advisor findings. The page presents charts and statistics that provide overall workload performance and query execution time potential improvement for the recommendations. You can use the page to show statement counts and recommendation action counts.

To see other aspects of the results for the advisor task, click one of the three other tabs on the page: Recommendations, SQL Statements, or Details.

The Recommendations page displays a chart and a table that show the top recommendations ordered by their percentage improvement to the total cost of the entire workload. The top recommendations have the biggest total performance improvement.

By clicking the Show SQL button, you can see the generated SQL script for the selected recommendations. You can click the corresponding recommendation identifier in the table to see the list of actions that need to be performed in order to implement the recommendation. On the Actions page, you can actually see all the corresponding SQL statements to execute in order to implement the action. For recommendations that you do not want to implement, keep those check boxes deselected. Then, click the Schedule Implementation button to implement the retained actions. This step is executed in the form of a Scheduler job.

SQL Performance Analyzer: Overview

- Targeted users: DBAs, QAs, application developers
- Helps predict the impact of system changes on SQL workload response time
- Builds different versions of SQL workload performance (that is, SQL execution plans and execution statistics)
- Executes SQL serially (concurrency not honored)
- Analyzes performance differences
- Offers fine-grained performance analysis on individual SQL
- Is integrated with SQL Tuning Advisor to tune regressions

ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Performance Analyzer: Overview

Oracle Database 11g includes SQL Performance Analyzer, which gives you an exact and accurate assessment of the impact of change on the SQL statements that make up the workload. SQL Performance Analyzer helps you forecast the impact of a potential change on the performance of a SQL query workload. This capability provides DBAs with detailed information about the performance of SQL statements, such as before-and-after execution statistics, and statements with performance improvement or degradation. This enables you (for example) to make changes in a test environment to determine whether the workload performance will be improved through a database upgrade.

SQL Performance Analyzer: Use Cases

SQL Performance Analyzer is beneficial in the following use cases:

- Database upgrades
- Implementation of tuning recommendations
- Schema changes
- Statistics gathering
- Database parameter changes
- OS and hardware changes

Accessible through Enterprise Manager and the DBMS_SQLPA package



ORACLE

Copyright © 2009, Oracle. All rights reserved.

SQL Performance Analyzer: Use Cases

SQL Performance Analyzer can be used to predict and prevent potential performance problems for any database environment change that affects the structure of the SQL execution plans. The changes can include (but are not limited to) any of the following:

- Database upgrades
- Implementation of tuning recommendations
- Schema changes
- Statistics gathering
- Database parameter changes
- OS and hardware changes

You can use SQL Performance Analyzer to predict SQL performance changes that result from changes for even the most complex environments. As applications evolve through the development life cycle, database application developers can test changes to schemas, database objects, and rewritten applications to mitigate any potential performance impact.

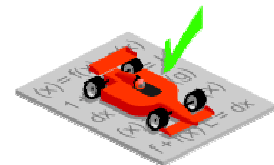
SQL Performance Analyzer also enables the comparison of SQL performance statistics.

You can access SQL Performance Analyzer through Enterprise Manager or by using the DBMS_SQLPA package.

For more information, see the *Oracle Database 11g: Performance Tuning* and the *Oracle Database 11g: New features for Administrators* course. For details about the DBMS_SQLPA package, see the *Oracle Database PL/SQL Packages and Types Reference Guide*.

Using SQL Performance Analyzer

1. Capture SQL workload on production.
2. Transport the SQL workload to a test system.
3. Build “before-change” performance data.
4. Make changes.
5. Build “after-change” performance data.
6. Compare results from steps 3 and 5.
7. Tune regressed SQL.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using SQL Performance Analyzer

1. **Gather SQL:** In this phase, you collect the set of SQL statements that represent your SQL workload on the production system.
2. **Transport:** You must transport the resultant workload to the test system. The STS is exported from the production system and the STS is imported into the test system.
3. **Compute “before-version” performance:** Before any changes take place, you execute the SQL statements, collecting baseline information that is needed to assess the impact that a future change might have on the performance of the workload.
4. **Make a change:** After you have the before-version data, you can implement your planned change and start viewing the impact on performance.
5. **Compute “after-version” performance:** This step takes place after the change is made in the database environment. Each statement of the SQL workload runs under a mock execution (collecting statistics only), collecting the same information as captured in step 3.
6. **Compare and analyze SQL Performance:** After you have both versions of the SQL workload performance data, you can carry out the performance analysis by comparing the after-version data with the before-version data.
7. **Tune regressed SQL:** At this stage, you have identified exactly which SQL statements may cause performance problems when the database change is made. Here, you can use any of the database tools to tune the system. After implementing any tuning action, you should repeat the process to create a new after-version and analyze the performance differences to ensure that the new performance is acceptable.

Quiz

Even when you enable Automatic Maintenance tasks, the SQL Tuning Advisor always has to be started separately.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Quiz

You can receive performance recommendations for historical SQL statements that are collected by AWR snapshots.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Quiz

The SQL Access Advisor can recommend the proper set of materialized views, materialized view logs, partitioning, and indexes for a given workload.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Quiz

The SQL Performance Analyzer provides you with detailed information about the performance of SQL statements, such as before-and-after execution statistics, and statements with performance improvement or degradation.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Summary

In this lesson, you should have learned how to:

- Use the SQL Tuning Advisor to:
 - Identify SQL statements that are using the most resources
 - Tune SQL statements that are using the most resources
- Use the SQL Access Advisor to tune a workload

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 15 Overview: Managing Performance by SQL Tuning

This practice covers the following topics:

- Using SQL Tuning Advisor

ORACLE

Copyright © 2009, Oracle. All rights reserved.

16

Managing Resources

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Configure the Database Resource Manager
- Access and create resource plans
- Create consumer groups
- Specify directives for allocating resources to consumer groups
- Map consumer groups to plans
- Activate a resource plan
- Monitor the Resource Manager

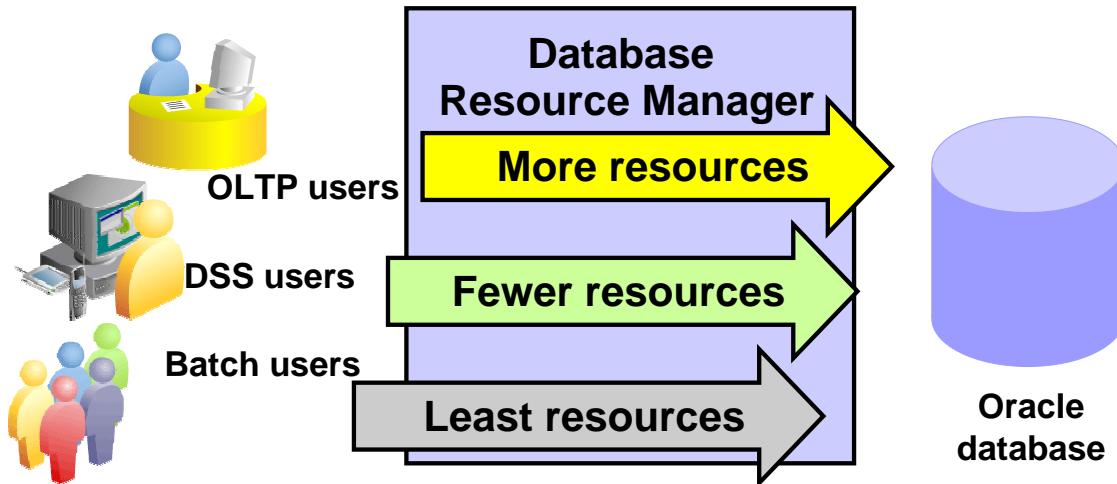
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Resource Manager: Overview

Use the Resource Manager to:

- Manage mixed workload
- Control system performance



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Resource Manager: Overview

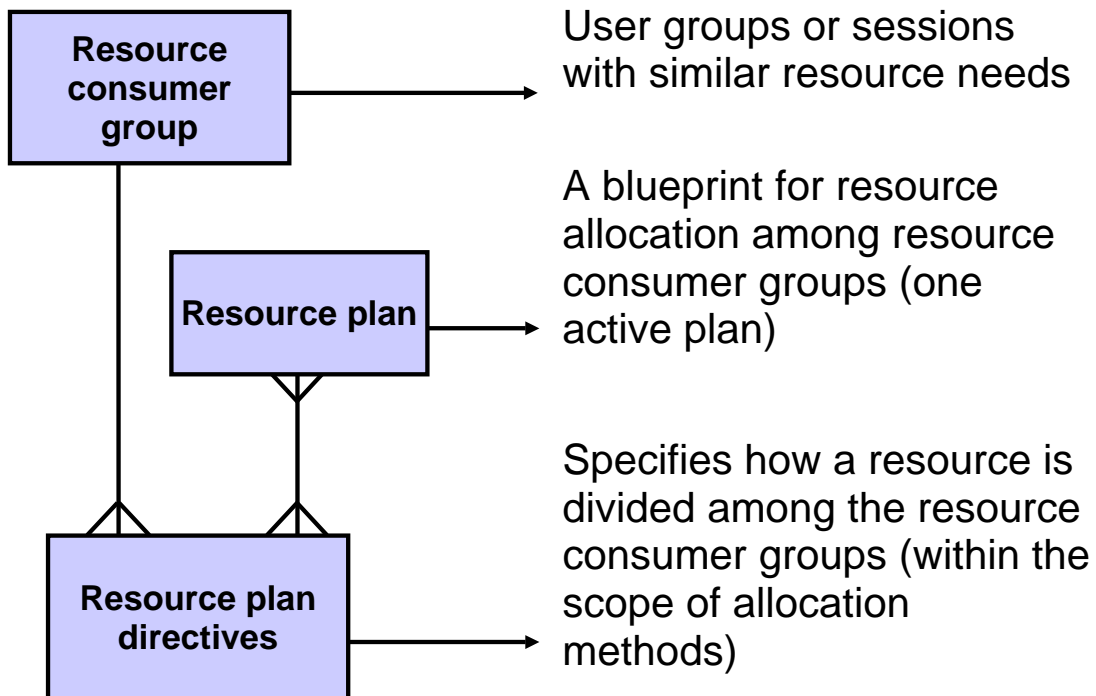
By using the Database Resource Manager (also called the Resource Manager), you have more control over the allocation of machine resources than is normally possible through operating system resource management alone. If resource management decisions are made by the operating system, it can lead to problems such as:

- Excessive overhead resulting from operating system context switching of Oracle database server processes when the number of server processes is high
- Suspension of a database server process that is holding a latch
- Unequal distribution of resources among all Oracle database processes, and an inability to prioritize one task over another
- Inability to manage database-specific resources, such as parallel execution servers and active sessions

The Database Resource Manager controls the distribution of resources among various sessions by controlling the execution schedule inside the database. By controlling which sessions run and for how long, the Database Resource Manager can ensure that resource distribution matches the plan directive and, therefore, the business objectives. With the Database Resource Manager, you can guarantee groups of users a minimum amount of processing resources regardless of the load on the system and the number of users.

The `DBMS_RESOURCE_MANAGER_PRIVS` package contains the procedures to grant and revoke the `ADMINISTER_RESOURCE_MANAGER` system privilege, which is a prerequisite for invoking the Resource Manager.

Database Resource Manager: Concepts



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Resource Manager: Concepts

Administering systems by using the Database Resource Manager involves the use of resource plans, resource consumer groups, and resource plan directives.

A *resource consumer group* defines a set of users or sessions that have similar requirements for using system and database resources.

A *resource plan* specifies how the resources are distributed among various resource consumer groups. The Database Resource Manager also allows for creation of plans within plans, called *subplans*.

Resource plan directives specify how a particular resource is shared among consumer groups or subplans. You associate resource consumer groups and subplans with a particular resource plan through plan directives.

Resource allocation methods determine what policy to use when allocating for any particular resource. Resource allocation methods are used by resource plans and resource consumer groups.

Why Use Resource Manager

- You can manage database and operating system resources, such as:
 - CPU usage
 - Degree of parallelism
 - Number of active sessions
 - Undo generation
 - Operation execution time
 - Idle time
 - Database consolidation
 - Server consolidation
- You can also specify criteria that, if met, cause the automatic switching of sessions to another consumer group.

Access via:

- EM



- DBMS_RESOURCE_MANAGER package

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Why Use Resource Manager

The Database Resource Manager provides several means of allocating resources:

- **CPU Method:** Enables you to specify how CPU resources are allocated among consumer groups and subplans
- **Degree of Parallelism Limit:** Enables you to control the maximum degree of parallelism for any operation within a consumer group
- **Active Session Pool with Queuing:** Allows you to limit the number of concurrent active sessions for a consumer group or subplan. If a group exceeds the maximum allowed number of sessions, new sessions are placed in a queue where they wait for an active session to complete. You can also specify a time limit on how long a session will wait before exiting with an error.
- **Undo Pool:** Enables you to control the total amount of undo that can be generated by a consumer group or subplan. Whenever the total undo space exceeds the amount specified by UNDO_POOL, no further INSERT, UPDATE, or DELETE commands are allowed until undo space is freed by another session in the same group or the undo pool is increased for the consumer group. If the consumer group's quota is exceeded during the execution of a DML statement, the operation aborts and returns an error. Queries are still allowed, even if a consumer group has exceeded its undo threshold.

Why Use Resource Manager (continued)

- **Execution Time Limit:** Allows you to specify a maximum execution time allowed for an operation. The Oracle database uses cost-based optimizer statistics to estimate how long an operation will take. If it is longer than the maximum time allowed (`MAX_EST_EXEC_TIME`), the operation returns an error and is not started. If a resource consumer group has more than one plan directive with `MAX_EST_EXEC_TIME` specified, the Resource Manager chooses the most restrictive of all incoming values.
- **Idle Time Limit:** Enables you to specify an amount of time for which a session can be idle, after which it will be terminated (`MAX_IDLE_TIME`). You can further restrict the Resource Manager to terminate only those sessions that are blocking other sessions (`MAX_IDLE_TIME_BLOCKER`).
- **Consumer Group Switching:** The initial consumer group is the group that a session would be in had it just logged in. The top call is defined as treating an entire PL/SQL block as one call or, similarly, treating SQL statements that are issued separately by the client as separate calls. This functionality is mostly beneficial for three-tier applications where the middle-tier server implements session pooling. In this case, the middle tier tends to do one call for an end user and then use the same session for a call for a different end user. Therefore, the boundaries of work are really calls, and the actions of a prior end user should not affect the next end user. You can create a plan directive, so that the Resource Manager automatically switches the user back to the initial consumer group at the end of the top call.

Note: You cannot specify both the `SWITCH_TIME_IN_CALL` and `SWITCH_TIME` parameters within the same directive. The `SWITCH_TIME` parameter is primarily intended for client/server applications, whereas the `SWITCH_TIME_IN_CALL` parameter is for three-tier applications.

- **Database Consolidation:** The Resource Manager enables you to optimize resource allocation among concurrent database sessions. Database consolidation requires that applications are isolated from each other. If one application experiences an increase in workload, that increase should not affect other applications. In addition, the performance of each application should be consistent. Good candidate applications for Database Consolidation are automated maintenance tasks because currently these applications can take up to 100% of the server CPU resources.
- **Server Consolidation:** Because many test, development, and small production databases are unable to fully utilize the servers that they are on, *server consolidation* provides a possible alternative. With server consolidation, resources are more fully utilized by running multiple database instances on the server. The method for managing CPU allocations on a multi-CPU server with multiple database instances is called Instance Caging. Because Instance Caging is simple to configure and does not require any new software to be licensed or installed, it is an excellent alternative to other server consolidation tools, such as virtualization and O/S workload managers.

You can access resource plans with the graphical interface of Enterprise Manager or the command line of the `DBMS_RESOURCE_MANAGER` package.

Default Maintenance Resource Manager Plan

```
SQL> show parameter resource_manager_plan
```

NAME	TYPE	VALUE
resource_manager_plan	string	SCHEDULER[0x2843]: DEFAULT_MAINTENANCE_PLAN

Group/Subplan	Level 1	Level 2
ORA\$AUTOTASK_SUB_PLAN	0	25
ORA\$DIAGNOSTICS	0	5
OTHER_GROUPS	0	70
SYS_GROUP	100	0

Group/Subplan	Level 1	Level 2	Level 3
ORA\$AUTOTASK_HIGH_SUB_PLAN	0	100	0
ORA\$AUTOTASK_MEDIUM_GROUP	0	0	100
ORA\$AUTOTASK_URGENT_GROUP	100	0	0

Group/Subplan	Percentage
ORA\$AUTOTASK_HEALTH_GROUP	25
ORA\$AUTOTASK_SPACE_GROUP	25
ORA\$AUTOTASK_SQL_GROUP	25
ORA\$AUTOTASK_STATS_GROUP	25

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Default Maintenance Resource Manager Plan

The automated maintenance tasks rely on the Resource Manager being enabled during the maintenance windows. When a maintenance window opens, the `DEFAULT_MAINTENANCE_PLAN` resource manager plan is automatically set to control the amount of CPU used by automated maintenance tasks. To be able to give different priorities to each possible task during a maintenance window, various consumer groups are assigned to `DEFAULT_MAINTENANCE_PLAN`. The hierarchy between groups and plans is shown in the slide.

For high priority tasks, see their group assignment:

- Optimizer Statistics Gathering is in the `ORA$AUTOTASK_STATS_GROUP` consumer group.
- Segment Advisor automatic is in the `ORA$AUTOTASK_SPACE_GROUP` consumer group.
- Automatic SQL Tuning is in the `ORA$AUTOTASK_SQL_GROUP` consumer group.

Note: If needed, you can change the percentage of CPU resources allocated to the various automated maintenance task consumer groups inside the `ORA$AUTOTASK_HIGH_SUB_PLAN`.

Example: DEFAULT_PLAN

Resource Consumer Group	Allocation Methods		
	MGMT_P1	MGMT_P2	MGMT_P3
SYS_GROUP	100%	0%	0%
OTHER_GROUPS	0%	90%	0%
ORA\$AUTOTASK_SUB_PLAN	0%	5%	0%
ORA\$DIAGNOSTICS	0%	5%	0%

For automated maintenance tasks

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Example: DEFAULT_PLAN

The DEFAULT_PLAN resource plan is one of the default plans provided for you. It contains directives for the following provided consumer groups:

- **SYS_GROUP:** The initial consumer group for the SYS and SYSTEM users
- **OTHER_GROUPS:** Used for all sessions that belong to consumer groups that are not part of the active resource plan. There must be a plan directive for OTHER_GROUPS in any active plan.
- **ORA\$AUTOTASK_SUB_PLAN:** A group with lower priority than SYS_GROUP and OTHER_GROUPS in this plan
- **ORA\$DIAGNOSTICS:** A group in this plan with the same priority as the ORA\$AUTOTASK_SUB_PLAN. The low priority of the ORA\$ groups prevents any automated maintenance work from consuming excessive amounts of system resources.

The initial consumer group of a user is the consumer group to which any session created by that user initially belongs. If you have not set the initial consumer group for a user, the user's initial consumer group will automatically be DEFAULT_CONSUMER_GROUP.

The DEFAULT_PLAN and associated resource consumer groups can be used or not used. It can be a template for new resource plans; it can be modified or deleted. Use it as appropriate for your environment.

Potential Work Flow

Your work flow for mandatory Resource Manager objects:

- Creating a new resource plan
- Creating a consumer group
- Assigning users to groups
- Specifying resource plan directives
- Activating a resource plan

```
DBMS_RESOURCE_MANAGER.CREATE  
_CONSUMER_GROUP(  
  CONSUMER_GROUP => 'APPUSER',  
  MGMT_MTH => 'ROUND-ROBIN',  
  COMMENT => '' );
```

```
DBMS_RESOURCE_MANAGER_PRIVS.  
GRANT_SWITCH_CONSUMER_GROUP  
(  
  grantee_name => 'PM',  
  consumer_group => 'APPUSER',  
  grant_option => FALSE );
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Potential Work Flow

To create a new plan, you need to configure several Resource Manager objects.

Creating a New Resource Plan

The Scheduler can automatically change the Resource Manager plan at the Scheduler window boundaries. Deselect the default Automatic Plan Switching Enabled check box if this option is unacceptable.

Creating Consumer Groups

Use the Resource Consumer Groups page in EM to create or edit a consumer group and description, to add or delete its users (members) and to define or edit its database roles.

You specify a resource allocation method for the distribution of CPU among sessions in the consumer group. “Round Robin” scheduling ensures that sessions are fairly executed. Therefore, the default allocation method is Round Robin. The “Run to Completion” allocation method specifies that sessions with the largest active time are scheduled ahead of other sessions. The equivalent functionality is achieved by the `DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP` procedure with the `MGMT_MTH` option.

Potential Work Flow (continued)

Assigning Users to Groups

- Users should be assigned to resource consumer groups. The user's default consumer group is the one to which any session created by that user initially belongs. If it is not set for a user, the user's initial consumer group defaults to `DEFAULT_CONSUMER_GROUP`. You must directly grant to the user, or to `PUBLIC`, the switch privilege to a consumer group before it can be the user's default consumer group. The switch privilege cannot come from a role granted to that user.
- The `DBMS_RESOURCE_MANAGER_PRIVS` package contains the procedure to assign resource consumer groups to users. Granting the switch privilege to a user enables the user to switch to a different consumer group.

Specifying Resource Plan Directives

Edit Resource Plan: DEFAULT_PLAN

1 2 3 4 5 6 Show SQL Revert Apply

General Parallelism Session Pool Undo Pool Threshold Idle Time

A Resource Plan contains directives that specify how resources are allocated to Consumer Groups. For each Consumer Group, a directive specifies the amount of CPU resources are allocated. It also specifies limits, such as the maximum degree of parallelism, execution time, and amount of I/O, that each session in the Consumer Group can consume. You can enable a Resource Plan manually or automatically, using Scheduler Windows.

Plan **DEFAULT_PLAN**

Description Default, basic, pre-defined plan that prioritizes SYS_GROUP

☐ Activate this plan

☐ Automatic Plan Switching Enabled

Resource Allocations

Mode: ☐ Percentage ☒ Advanced

Group/Subplan	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8
APPUSER			60					
LOW_GROUP			40					
ORA\$AUTOTASK_SUB_PLAN		5						
ORA\$DIAGNOSTICS		5						
OTHER_GROUPS		90						
SYS_GROUP	100							

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying Resource Plan Directives

If you do not use Enterprise Manager to create the resource plan or resource consumer groups, you must first create a *pending area*. This is a scratch area that enables you to stage your changes and to validate them before they are made active.

In Enterprise Manager, there are several property pages, which you can use for specifying plan directives:

1. On the General page, associate consumer groups with plans and specify how much CPU each consumer group or subplan gets with the MGMT_MTH value.
2. Specify a parallel degree limit to control the maximum degree of parallelism for any operation within a consumer group.
3. You can control the maximum number of concurrently active sessions allowed within a consumer group. An entire parallel execution session is counted as one active session.
4. You can control the amount of total undo that can be generated by a consumer group.
5. You can specify threshold values, such as execution time limit (in seconds), I/O limit (in MB), and I/O request limit (in number of requests).
6. You can specify an amount of time that a session can be idle, after which it will be terminated. You can further restrict such termination to only those sessions that are blocking other sessions.

Note: The following slides provide further details about directives using the tab numbers indicated in this slide. Refer to this slide when you see “Directive Tab *n*” on subsequent slides.

Resource Allocation Methods for Resource Plans

Parameter (Comments)	Possible Values
MGMT_MTH	EMPHASIS, RATIO
Allocating CPU usage	
PARALLEL_DEGREE_LIMIT_MTH	PARALLEL_DEGREE_LIMIT_ABSOLUTE
Limiting degree of parallelism of any operation	
ACTIVE_SESS_POOL_MTH	PARALLEL_DEGREE_LIMIT_ABSOLUTE
Limiting number of active sessions, queuing inactive ones	
QUEUING_MTH	FIFO_TIMEOUT
Controlling queues, how inactive sessions enter active session pool	

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Resource Allocation Methods for Resource Plans

Resource allocation methods determine how the Resource Manager allocates a particular resource to a resource consumer group or resource plan. You specify values for the following resource allocation methods when creating the resource plan.

There are two ways of specifying the CPU distribution with the MGMT_MTH parameter:

- EMPHASIS is the default method for single-level plans. It is also used for multilevel plans that use percentages to specify how CPU is distributed among consumer groups.
- RATIO is for single-level plans that use ratios to specify how CPU is distributed.

PARALLEL_DEGREE_LIMIT_MTH limits the maximum degree of parallelism of any operation.

This method can be specified only for resource consumer groups, not subplans. The PARALLEL_DEGREE_LIMIT_ABSOLUTE method is the only possible value, specifying how many processes may be assigned to an operation. If there are multiple plan directives referring to the same subplan or consumer group, the **minimum** of all the possible values is used as the parallel degree limit for that subplan or consumer group.

The ACTIVE_SESS_POOL_MTH parameter limits the number of active sessions. All other sessions are inactive and wait in a queue to be activated. The only value (that is, the only available method) for this parameter is PARALLEL_DEGREE_LIMIT_ABSOLUTE, which is its default value.

QUEUING_MTH controls the order in which queued inactive sessions execute. FIFO_TIMEOUT is the default and only method available.

Comparison of EMPHASIS and RATIO

EMPHASIS	RATIO
The value specifies the maximum percentage of CPU resources a consumer group can use.	The value specifies a number that indicates the ratio of CPU resources to be allocated to the consumer group.
You can allocate resources for up to 8 different levels.	You can specify values for only one level.
The sum of percentages at any given level must be less than or equal to 100.	You must use integer values, but there is no limit on the sum of values.
Default value is NULL.	Default value is NULL.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Comparison of EMPHASIS and RATIO

The CPU allocation method **EMPHASIS** determines how much emphasis is given to sessions in different consumer groups in a resource plan. CPU usage is assigned levels from 1 through 8, with Level 1 having the highest priority. Percentages specify how to allocate CPU to each consumer group at each level.

The following rules apply for the **EMPHASIS** resource allocation method:

- CPU resources are distributed at a given level on the basis of the specified percentages. The percentage of CPU specified for a resource consumer group is a maximum for how much that consumer group can use at a given level.
- Consumer resources that are not used at a given level are made available to consumer groups at the next level. For example, if the consumer groups at Level 1 use only 60% of the available resources, the additional 40% is made available to consumer groups at Level 2.
- The sum of percentages at any given level must be less than or equal to 100.
- Any levels that have no plan directives explicitly specified have a default of 0% for all subplans or consumer groups.
- The **EMPHASIS** resource allocation method avoids starvation problems, where consumers with lower priorities are not given the opportunity to run.

Comparison of EMPHASIS and RATIO (continued)

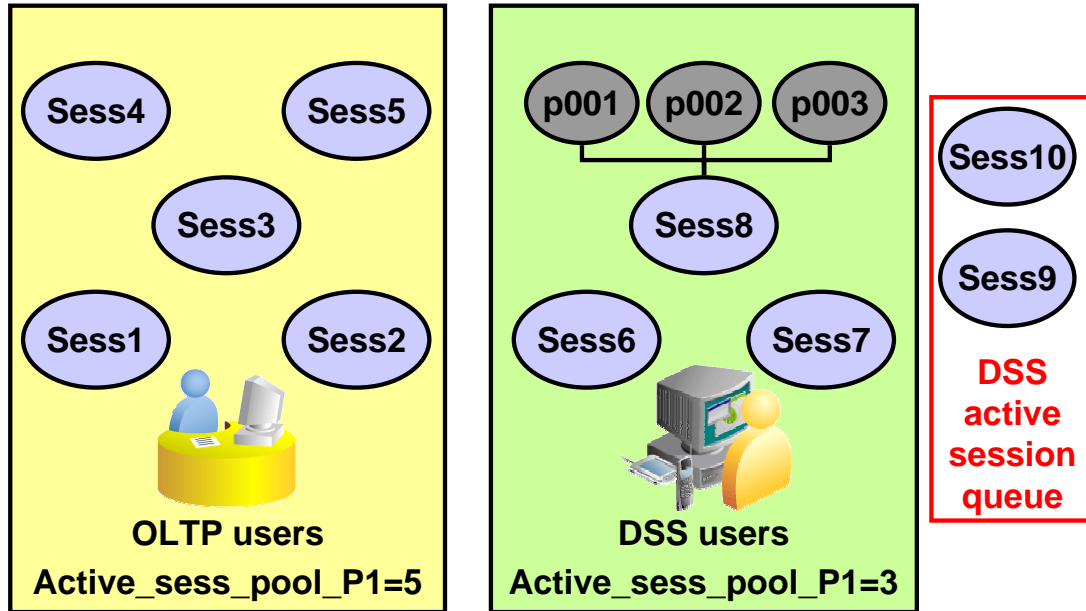
The RATIO policy is a single-level CPU allocation method. Instead of percentages, you specify numbers corresponding to the ratio of CPU you want to give to the consumer group. For example, given three consumer groups OLTP_USERS, DSS_USERS, and BATCH_USERS, you can specify the following ratios:

- OLTP_USERS: 4
- DSS_USERS: 3
- BATCH_USERS: 2
- OTHER: 1

This is similar to saying that OLTP users should get 40% of the resources, DSS users should get 30% of the resources, BATCH users should get 20% of the resources, and all other consumer groups should get 10% of the available resources.

If there are no consumers in the OTHER or DSS_USERS consumer groups currently utilizing CPU resources, then the OLTP_USERS consumer group would get two-thirds of the available resources and the BATCH_USERS consumer group would get the other third.

Active Session Pool Mechanism



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Active Session Pool Mechanism

Using the Active Session Pool feature, you can control the maximum number of concurrently active sessions per resource consumer group. With this functionality, a DBA can indirectly control the amount of resources that any resource consumer group uses because resource consumption is proportional to the number of active sessions. Using an active session pool can help to reduce the number of servers taking resources in the system, thus avoiding inefficient paging, swapping, and other resource depletion (such as memory) resulting from attempting to run too many jobs simultaneously.

After the Active Session Pool is filled with active sessions, the Resource Manager queues all subsequent sessions attempting to become active until other active sessions complete or become inactive. An active session is one currently involved in a transaction, query, or parallel operation. Individual parallel slaves are not counted as sessions; the entire parallel operation counts as one active session.

There is only one queue per resource consumer group and the queuing method is first in, first out (FIFO) with a timeout. The queue is implemented as a memory structure and cannot be queried directly.

Setting the Active Session Pool

Edit Resource Plan: DEFAULT_PLAN

Actions

[General](#) [Parallelism](#) **Session Pool** [Undo Pool](#) [Threshold](#) [Idle Time](#)

Specify a limit on the maximum number of concurrently active sessions for a consumer group. All other sessions will wait in an activation queue.

Group	Maximum Number of Active Sessions	Activation Queue Timeout (sec)
APPUSER	50	UNLIMITED
LOW_GROUP	UNLIMITED	UNLIMITED
ORA\$DIAGNOSTICS	UNLIMITED	UNLIMITED
OTHER_GROUPS	UNLIMITED	UNLIMITED
SYS_GROUP	UNLIMITED	UNLIMITED



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Setting the Active Session Pool

You can easily configure the Active Session Pool settings for a resource plan by using Enterprise Manager.

Example, if you limit the Maximum Number of Active Sessions to 50 for the APPUSER consumer group:

```
BEGIN
dbms_resource_manager.clear_pending_area();
dbms_resource_manager.create_pending_area();
dbms_resource_manager.update_plan_directive(
    plan => 'DEFAULT_PLAN',
    group_or_subplan => 'APPUSER',
    new_comment => '',
    new_active_sess_pool_p1 => 50,
    new_queueing_p1 => NULL,
    new_parallel_degree_limit_p1 => NULL,
    new_switch_group => '',
    new_switch_time => NULL,
    new_switch_estimate => false,
```

Setting the Active Session Pool (continued)

```
new_max_est_exec_time => NULL,  
new_undo_pool => NULL,  
new_max_idle_time => NULL,  
new_max_idle_blocker_time => NULL,  
mgmt_p1 => NULL,  
mgmt_p2 => NULL,  
mgmt_p3 => 60,  
mgmt_p4 => NULL,  
mgmt_p5 => NULL,  
mgmt_p6 => NULL,  
mgmt_p7 => NULL,  
mgmt_p8 => NULL,  
switch_io_megabytes => NULL,  
switch_io_reqs => NULL,  
switch_for call);  
dbms_resource_manager.submit_pending_area();  
END;
```

Specifying Thresholds

Specifying execution time limit:

- Proactive estimation of the execution time for an operation (via cost-based optimizer statistics), default: `UNLIMITED`
- Specifying maximum estimated execution time at the resource consumer group level
- No start allowance for huge jobs, if the estimate is longer than `MAX_EST_EXEC_TIME`: (ORA-07455)

Specifying other thresholds:

- Limiting session I/O with `SWITCH_IO_MEGABYTES` (in MB)
- Limiting session I/O requests with `SWITCH_IO_REQS`

Returning to original consumer group with `SWITCH_FOR_CALL` (Default: `FALSE`, consumer group is not restored)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying Threshold

You can define the maximum estimated execution time any operation can take at any given time by setting the resource plan directive's `MAX_EST_EXEC_TIME` parameter.

- When this parameter is set, the Database Resource Manager estimates the time a specific job will take, which is calculated using the statistics from the cost-based optimizer.
- If a resource consumer group has more than one plan directive referring to it, it may have more than one `MAX_EST_EXEC_TIME`. The Database Resource Manager then chooses the most restrictive of all incoming values.
- If the operation's estimate is more than `MAX_EST_EXEC_TIME`, then the operation does not start and the ORA-07455 error is issued. This eliminates any exceptionally large jobs that would utilize too many system resources.
- The `SWITCH_IO_MEGABYTES` directive specifies the amount of I/O (in MB) that a session can issue before an action is taken. The default is `NULL`, which means unlimited.
- The `SWITCH_IO_REQS` directive specifies the number of I/O requests that a session can issue before an action is taken. The default is `NULL`, which means unlimited.
- The `SWITCH_FOR_CALL` directive specifies that if an action is taken because of the `SWITCH_TIME`, `SWITCH_IO_MEGABYTES`, or `SWITCH_IO_REQS` parameters, the consumer group is restored to its original consumer group at the end of the top call. Default is `FALSE`, which means that the original consumer group is not restored at the end of the top call.

Setting Idle Timeouts

Edit Resource Plan: DEFAULT_PLAN

Actions Create Like Go Show SQL Revert Apply

General Parallelism Session Pool Undo Pool Threshold Idle Time

Specify the maximum time a session in the consumer group can be idle.

Group	Max Idle Time (sec)	Max Idle Time if Blocking Another Session (sec)
APPUSER	600	300
LOW_GROUP	UNLIMITED	UNLIMITED
ORA\$DIAGNOSTICS	UNLIMITED	UNLIMITED
OTHER_GROUP		
SYS_GROUP		

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE
(PLAN => 'DAY_PLAN',
 GROUP_OR_SUBPLAN => 'APPUSER',
 COMMENT => 'Limit Idle Time Example',
 NEW_MAX_IDLE_TIME => 600,
 NEW_MAX_IDLE_BLOCKER_TIME => 300);
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Setting Idle Timeouts

You use the resource plan's Idle Time tab to set the maximum idle timeouts for a resource plan. "Max Idle Time (sec)" and "Max Idle Time if Blocking Another Session (sec)" are the respective equivalents of the `NEW_MAX_IDLE_TIME` and `NEW_MAX_IDLE_BLOCKER_TIME` resource directives in the `DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE` procedure. They are both specified in seconds.

`NEW_MAX_IDLE_TIME` specifies the time that a session is neither executing nor waiting for I/O. When the session exceeds the specified limit, the PMON process forcibly kills the session and cleans up its state. In addition to limiting the maximum idle time for a session, you can also limit the amount of time that an idle session can block another session. You impose this limit by setting the `NEW_MAX_IDLE_BLOCKER_TIME` resource directive to the number of seconds to allow a session to be idle while blocking another session. You can also specify a value of `UNLIMITED` to indicate that no maximum time has been set. The default is `NULL`, which means unlimited. These settings give you a more granular control than profiles, whose single value cannot distinguish between blocking and nonblocking sessions.

In the slide example, the PMON process kills sessions that are idle for longer than 600 seconds. The PMON process also kills sessions that are idle for more than 300 seconds and are blocking other sessions. PMON checks these limits once every minute and if it finds a session that has exceeded one of the limits, it forcibly kills the session and cleans up all its resources.

Limiting CPU Utilization at the Database Level

Database consolidation requirements:

- Applications isolated from each other
- Consistent performance

CPU directives can be used to:

- Specify a minimum CPU allocation for each application
- Designate how unused allocations should be redistributed
- Specify the `MAX_UTILIZATION_LIMIT` attribute to impose an absolute upper limit on CPU utilization (which overrides any redistribution of CPU within a plan)
- Good candidate: Auto-maintenance tasks

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Limiting CPU Utilization at the Database Level

For concurrent database sessions: Database consolidation requires that applications are isolated from each other. If one application experiences an increase in workload, that increase should not affect other applications. In addition, the performance of each application should be consistent.

Fixed Policy CPU Resource Management

The `MAX_UTILIZATION_LIMIT` attribute of resource plan directives enables you to impose an absolute upper limit on CPU utilization for a resource consumer group. This absolute limit overrides any redistribution of CPU within a plan.

Note: Good candidate applications for database consolidation are automated maintenance tasks because currently these applications can take up to 100% of the server CPU resources. You can set a maximum limit for each auto-task consumer group.

Limiting CPU Utilization at the Database Level

Specify minimum and maximum CPU utilization limits.

<u>DB Consolidation Plan #1</u>		
	CPU Allocation	Maximum Utilization Limit
App 1	50%	60%
App 2	20%	30%
App 3	20%	30%
App 4	10%	20%

Specify maximum CPU utilization limits only.

<u>DB Consolidation Plan #2</u>		
	CPU Allocation	Maximum Utilization Limit
App 1	null	50%
App 2	null	20%
App 3	null	20%
App 4	null	10%

```
EXEC DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE( -  
  plan                => 'db_consolidation_plan',  
  group_or_subplan    => 'App_1',  
  mgmt_pl             => 50,  
  max_utilization_limit => 60);
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Limiting CPU Utilization at the Database Level (continued)

The MAX_UTILIZATION_LIMIT directive limits the CPU consumption of an application. You can set minimum and maximum boundaries, as shown in the slide.

The PL/SQL example in the slide specifies a minimum value percentage (50%) for the CPU allocation resource at level 1 for the APP_1 consumer group. This example also specifies an absolute maximum CPU utilization percentage (60%) permitted for that same consumer group. The example uses the DB_CONSOLIDATION_PLAN plan.

Similar commands can be executed for each consumer group shown in the sample tables.

Note: In releases prior to Oracle Database 11gR2, the implicit maximum utilization limit was set to 100%.

Limiting CPU Utilization at the Server Level: Instance Caging

- Managing CPU allocations on a multi-CPU server with multiple database instances
- Enabling instance caging :
 - Enable any CPU resource plan.

```
alter system set resource_manager_plan = 'default_plan';
```

- Specify the maximum number of CPUs that the instance can use at any time.

```
alter system set cpu_count=4;
```

Two approaches:

- Over-provisioning: The sum of the CPU limit for each instance exceeds the actual number of CPUs.
- Partitioning: The sum of the CPU limit for each instance equals the actual number of CPUs.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Limiting CPU Utilization at the Server Level: Instance Caging

Because many test, development, and small production databases are unable to fully utilize the servers that they are on, *server consolidation* provides a possible alternative. With server consolidation, resources are more fully utilized by running multiple database instances on the server. However, this may bring about CPU contention and an adverse impact due to workload surges on one instance.

Instance caging is a method that uses the CPU_COUNT initialization parameter to limit the number of CPUs that an instance can use. In addition, the Resource Manager is employed to allocate the CPUs for the database sessions based on the instance resource plan.

Configure instance caging in two steps, by enabling:

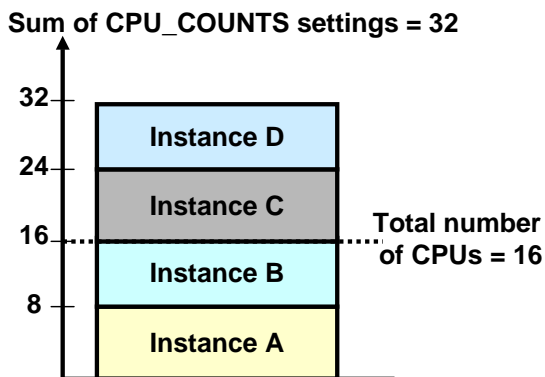
- The Resource Manager, which limits the amount of CPU that the database instance consumes
- The CPU_COUNT parameter, which specifies the maximum (the limit), not actual amount of CPU that the database instance can use at any time

By default, the CPU Resource Manager assumes that the database instance can use all CPUs on a server. To enable instance caging, any resource plan with CPU directives can be used.

Instance Caging Examples

Over-provisioning approach:

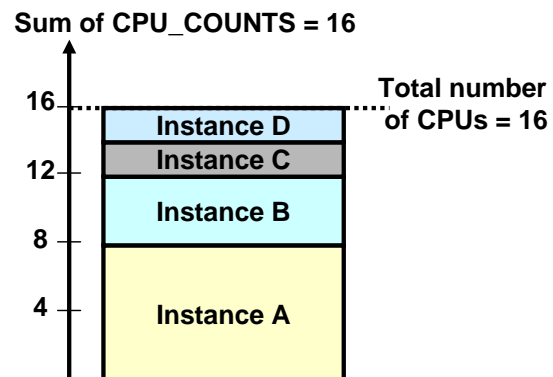
One database instance can still impact the others.



With all four instances active, one instance can get $4 / (4 + 4 + 4 + 4) = 25\%$ of CPU.

Partitioning approach:

One database instance cannot impact the others.



Each instance has a dedicated number of CPUs.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Instance Caging Examples

Over-provisioning approach: This approach is appropriate for noncritical databases and low-load, noncritical production systems. Although the instances impact each other's performance, at any given time, one or more of the instances may be idle or experiencing a low load.

Although the database instances can impact each other's performance, instance caging limits their impact and helps to provide predictable performance. In the left example, where all four instances have CPU_COUNT set to 4, the maximum percentage of CPU that a database instance can consume at any point in time is its own limit divided by the sum of the limits for all active databases. In this example, one instance will be able to consume $4 / (4 + 4 + 4 + 4) = 25\%$ of the CPU. If only two instances are active, one instance will be able to consume $4 / (4 + 4) = 50\%$ of the CPU.

Partitioning approach: This approach is appropriate for critical product systems. It prevents the instances from interfering with each other and provides predictable performance.

Instance caging can partition the CPU resources by ensuring that the sum of all CPU limits does not exceed the total number of CPUs. In the example on the right, if four database instances share a 16-CPU server, their limits can be set to 8, 4, 2, and 2. By dedicating CPU resources to a database instance, partitioning provides two advantages:

- One database instance's CPU load cannot affect another's.
- Each database instance's CPU resources is fixed, leading to more predictable performance.

Monitoring Instance Caging

View value of the CPU_COUNT parameter:

```
SELECT value FROM v$parameter WHERE name = 'cpu_count'
AND (isdefault = 'FALSE' OR ismodified != 'FALSE');
```

Determine the Resource Manager status:

```
SELECT name FROM v$rsrc_plan
WHERE is_top_plan = 'TRUE' AND cpu_managed = 'ON';
```

Manage throttling:

```
SELECT begin_time, consumer_group_name,
       cpu_consumed_time, cpu_wait_time
FROM v$rsrcmgrpmetric_history
ORDER BY begin_time;
```

```
SELECT name, consumed_cpu_time, cpu_wait_time
FROM v$rsrc_consumer_group;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring Instance Caging

- If the CPU_COUNT parameter is not set, no value is returned by the first query.
- If no rows are returned by the second query in the slide, the Resource Manager is not managing the CPU. If a row is returned, it indicates the active plan.

Instance caging limits the CPU consumption of the foreground processes by throttling them. A foreground process is throttled when it is waiting on the “resmgr:cpu quantum” wait event.

You can monitor the amount of throttling in two ways:

- The V\$RSRCMGRMETRIC_HISTORY view shows the amount of CPU consumption (CPU_CONSUMED_TIME) and throttling (CPU_WAIT_TIME) for each minute in the past hour. Values are displayed in milliseconds.
- The V\$RSRC_CONSUMER_GROUP view shows the amount of CPU consumption (CPU_CONSUMED_TIME) and throttling (CPU_WAIT_TIME) since CPU Resource Management was enabled. The time is displayed in milliseconds.

Note: For case studies, see the Oracle White Paper titled *Database Instance Caging: A Simple Approach to Server Consolidation*.

Resource Consumer Group Mapping

Consumer Group Mappings

General **Priorities** Show SQL Revert Apply

Create rules to enable the resource manager to automatically assign sessions to consumer groups.

View: All

Add Rule for Selected Type

Select	Priority	View	Value	Consumer Group
<input checked="" type="radio"/>	1	Service Module and Action	No Mappings Specified	
<input type="radio"/>	2	Service and Module	No Mappings Specified	
<input type="radio"/>	3	Module and Action	No Mappings Specified	
<input type="radio"/>	4	Module	No Mappings Specified	
<input type="radio"/>	5	Service	No Mappings Specified	
<input type="radio"/>	6	Oracle User	HR APPUSER SCOTT LOW_GRO SYS, SYSTEM SYS_GRO	
<input type="radio"/>	7	Client Program	No Mappings Specified	
<input type="radio"/>	8	Client OS User	ORACLE SYS_GRO	
<input type="radio"/>	9	Client Machine	No Mappings Specified	

Attribute Mappings

Reorder the list of mappings to set priorities. Mappings at the top of the list receive the highest priority. In order to decide between conflicting mappings, you can establish a priority ordering of the attributes from most important to least important. The priority of each attribute is set to a unique integer from 1 to 10.

Service Module and Action
Service and Module
Module and Action
Module
Service
Oracle User
Client Program
Client OS User
Client Machine

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Resource Consumer Group Mapping

You can configure the Database Resource Manager to automatically assign consumer groups to sessions by providing mappings between session attributes and consumer groups. Further, you can prioritize the mappings so as to indicate which mapping has precedence in case of conflicts. There are two types of session attributes: login attributes and run-time attributes. The login attributes (the last five in the Attribute Mappings list shown in the slide) are meaningful only at session login time, when the Database Resource Manager determines the initial consumer group of the session. In contrast, a session that has already logged in can later be reassigned to another consumer group on the basis of its run-time attributes.

From the Database Control home page, navigate to the Server tabbed page, and then click the Resource Consumer Group Mappings link in the Resource Manager section. For each of the attributes, set up a mapping that consists of a way to identify a session (for example, username), and a consumer group. Add or remove rows for each of the resource consumer group categories, as required, and enter text identifying the user, client, module, or service in the corresponding group. You can establish a priority ordering between conflicting mappings of the attributes by using the Priorities tab. You can set the priority from the most important to the least important by using the navigational arrows (as highlighted). The mappings at the top of the list have the highest priority. Using EM Database Control, you can easily view the SQL generated from your actions by clicking the Show SQL button.

Resource Consumer Group Mapping (continued)

Example to give the Client OS User a higher priority than the Client Program:

```
BEGIN
dbms_resource_manager.clear_pending_area();
dbms_resource_manager.create_pending_area();
dbms_resource_manager.set_consumer_group_mapping(
    dbms_resource_manager.oracle_user,
    'SCOTT',
    'LOW_GROUP'
);
dbms_resource_manager.set_consumer_group_mapping_pri(
    EXPLICIT => 1,  SERVICE_MODULE_ACTION => 2,
    SERVICE_MODULE => 3,
    MODULE_NAME_ACTION => 4,
    MODULE_NAME => 5,
    SERVICE_NAME => 6,
    ORACLE_USER => 7,
    CLIENT_OS_USER => 8,
    CLIENT_PROGRAM => 9,
    CLIENT_MACHINE => 10
);
dbms_resource_manager.submit_pending_area();
END;
```

Activating a Resource Plan

Edit View Delete Actions Activate Go				
Select	Plan	Status	Description	Scheduler Windows
<input checked="" type="radio"/>	DEFAULT MAINTENANCE PLAN		Default plan for maintenance windows that prioritizes SYS_GROUP operations and allocates the remaining 5% to diagnostic operations and 25% to automated maintenance operations.	MONDAY WINDOW TUESDAY WINDOW WEDNESDAY WINDOW THURSDAY WINDOW FRIDAY WINDOW SUNDAY WINDOW SATURDAY WINDOW

EM > Server > Settings (in the Resource Manager section)

ORACLE Enterprise Manager 11g Database Control [Setup](#) [Preferences](#) [Help](#) [Logout](#)

Database Instance: orcl > Logged in As SYS

Resource Manager Settings

Active Resource Plan **No Active Plan**

Available Resource Plans

Configure properties of Resource Management that apply to all Resource Plans.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Activating a Resource Plan

You can use the Plans page of Enterprise Manager to manage resource plans. To activate a plan, select the plan you want to make active, choose Activate from the Actions drop-down list, and then click Go. The plan you selected is then made the current top plan for the instance.

Using the **RESOURCE_MANAGER_PLAN** Initialization Parameter

The plan for an instance is defined using the **RESOURCE_MANAGER_PLAN** database initialization parameter. This parameter specifies the top plan to be used for this instance. If no plan is specified, the Resource Manager is not activated for the instance.

You can activate, deactivate, or change the current top plan by using an **ALTER SYSTEM** statement. When a resource plan is changed using this command, the change takes effect instantly.

If the parameter is set in a parameter file, and the plan specified is not defined in the database, then the database cannot be opened with that parameter file. The following error is returned:

```
ORA-07452: specified resource manager plan does not exist in the
data dictionary
```

If this error is encountered, the parameter must be modified to show a correct value before the instance can be restarted.

Database Resource Manager Information

View Name	Information
DBA_RSRC_PLANS	Plans and status
DBA_RSRC_PLAN_DIRECTIVES	Plan directives
DBA_RSRC_CONSUMER_GROUPS	Consumer groups
DBA_RSRC_CONSUMER_GROUP_PRIVS	Users/roles
DBA_RSRC_GROUP_MAPPINGS	Consumer group mapping
DBA_RSRC_MAPPING_PRIORITY	Mapping priority
DBA_USERS	Column initial_rsrc_consumer_group
DBA_RSRC_MANAGER_SYSTEM_PRIVS	Users/roles

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Resource Manager Information

Several data dictionary views are available to check the resource plans, consumer groups, and plan directives that are declared in the instance. This section discusses some useful information that can be obtained from these views. For more detailed information about the contents of each of these views, refer to the *Oracle Database Reference* manual.

Use the following query to obtain information about resource plans defined in the database:

```
SQL> SELECT plan, num_plan_directives, status, mandatory
2 FROM dba_rsrc_plans;
```

PLAN	NUM_PLAN_DIRECTIVES	STATUS	MAN
DEFAULT_PLAN	3	ACTIVE	NO
INTERNAL QUIESCE	2	ACTIVE	YES
INTERNAL_PLAN	1	ACTIVE	YES
BUGDB_PLAN	4	ACTIVE	NO
MAILDB_PLAN	3	ACTIVE	NO
MYDB_PLAN	3	ACTIVE	NO

A status of ACTIVE indicates that the plan has been submitted and can be used, whereas a status of PENDING shows that the plan has been created, but is still in the pending area.

If the mandatory column is assigned a value of YES, then the plan cannot be deleted.

Monitoring the Resource Manager

Queued Sessions							
Consumer Group	Configured Limits		Statistics				
	Active Session Limit	Maximum Time in Queue (sec)	Current Queued Sessions	Total Queued Sessions	Time in Queue (sec)	Queue Time	Time Outs
APPUSER	UNLIMITED	UNLIMITED	0	0	0	0	0
SYS_GROUP	UNLIMITED	UNLIMITED	0	0	0	0	0
OTHER_GROUPS	UNLIMITED	UNLIMITED	0	0	0	0	0
ORA\$DIAGNOSTICS	UNLIMITED	UNLIMITED	0	0	0	0	0
LOW_GROUP	UNLIMITED	UNLIMITED	0	0	0	0	0

Automatic Reprioritization							
Consumer Group	Configured Limits			Statistics			
	CPU Time Limit (ms)	I/O Requests	I/O (MB)	Switch into Group	Switch out of the Group	Active Sessions Killed	Calls Cancelled
APPUSER	UNLIMITED	UNLIMITED	UNLIMITED	0	0	0	0
SYS_GROUP	UNLIMITED	UNLIMITED	UNLIMITED	0	0	0	0
OTHER_GROUPS	UNLIMITED	UNLIMITED	UNLIMITED	0	0	0	0
ORA\$DIAGNOSTICS	UNLIMITED	UNLIMITED	UNLIMITED	0	0	0	0
LOW_GROUP	UNLIMITED	UNLIMITED	UNLIMITED	0	0	0	0

Idle					
Consumer Group	Idle Blocker Limit		Idle Limit		
	Maximum Idle Blocker Time (sec)	Blocker Sessions Killed	Maximum Idle Time (sec)	Idle Sessions Killed	
APPUSER	UNLIMITED	0	UNLIMITED	0	
SYS_GROUP	UNLIMITED	0	UNLIMITED	0	
OTHER_GROUPS	UNLIMITED	0	UNLIMITED	0	
ORA\$DIAGNOSTICS	UNLIMITED	0	UNLIMITED	0	
LOW_GROUP	UNLIMITED	0	UNLIMITED	0	

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring the Resource Manager

You can monitor the functioning of the Database Resource Manager at the session level. It is integrated with Automatic Database Diagnostic Monitor (ADDM).

There are different ways to manage and monitor the Resource Manager by using EM Database Control. On the Server tabbed page, click the Statistics link in the Resource Manager section.

The Resource Manager Statistics page displays a grouping of statistics and charts that depict the current state of the active resource plan. You can view the statistics for the currently active plan.

For Resource Usage, you can view “CPU Consumed,” “I/O Requests per Second,” and “Megabytes of I/O Issued per Second.” Another chart displays “Resource Manager Induced Waits.” Then there are statistics for “Queued Sessions,” “Automatic Reprioritization,” and idle time.

Monitoring the Resource Manager

- V\$SESSION: Contains the `resource_consumer_group` column that shows the current group for a session
- V\$RSRC_PLAN: A view that shows the active resource plan
- V\$RSRC_CONSUMER_GROUP: A view that contains statistics for all active groups

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Monitoring the Resource Manager (continued)

CPU Utilization

There are at least three different views in the system that can provide you with information about the CPU utilization inside the Oracle database:

- V\$RSRC_CONSUMER_GROUP shows CPU utilization statistics on a per consumer group basis, if you are running the Oracle Database Resource Manager. This view displays data related to currently active resource consumer groups.
- V\$SYSSTAT shows the Oracle database CPU usage for all sessions. The statistic “CPU used by this session” shows the aggregate CPU used by all sessions.
- V\$SESSTAT shows the Oracle database CPU usage per session. You can use this view to determine which particular session is using the most CPU.

The V\$RSRC_CONSUMER_GROUP View

The following is a quick description of some of the columns in this view:

- **name:** Name of the consumer group
- **active_sessions:** Number of currently active sessions in this consumer group
- **execution_waiters:** Number of active sessions waiting for a time slice
- **requests:** Cumulative number of requests executed in this consumer group
- **cpu_wait_time:** Cumulative amount of time that sessions waited for CPU
- **consumed_cpu_time:** Cumulative amount of CPU time consumed by all sessions

Monitoring the Resource Manager (continued)

There is no view that shows the Active Session Pool queue directly, but you can get some information from:

- **V\$SESSION:** The `current_queue_duration` column shows how long a session has been queued, or 0 (zero) if the session is not currently queued.
- **V\$RSRC_CONSUMER_GROUP:** The `queue_length` column shows the number of sessions currently queued per consumer group.

Quiz

Select the statements that are true about the Resource Manager and its functionality:

1. You can set threshold values only for execution time, not for session I/O.
2. You can limit CPU utilization at the database level to isolate applications for each other.
3. On a multi-CPU server with multiples database instances, you can limit each server's CPU utilization by enabling instance caging.
4. When the SWITCH_TIME, SWITCH_IO_MEGABYTES, or SWITCH_IO_REQS parameters cause a switch in consumer groups, you can never return to the original consumer groups.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 3

Summary

In this lesson, you should have learned how to do the following:

- Configure the Database Resource Manager
- Access and create resource plans
- Create consumer groups
- Specify directives for allocating resources to consumer groups
- Map consumer groups to plans
- Activate a resource plan
- Monitor the Resource Manager

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 16 Overview: Using the Resource Manager

This practice covers the following topics:

- Creating a resource consumer group
- Specifying CPU resource allocation directives for consumer groups
- Associating users with a resource consumer group
- Activating a resource plan
- Testing in SQL*Plus
- Deactivating a resource plan

ORACLE

Copyright © 2009, Oracle. All rights reserved.

17

Automating Tasks with the Scheduler

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Simplify management tasks by using the Scheduler
- Create a job, program, and schedule
- Monitor job execution
- Use a time-based or event-based schedule for executing Scheduler jobs
- Describe the use of windows, window groups, job classes, and consumer groups
- Use email notification
- Use job chains to perform a series of related tasks
- Describe Scheduler jobs on remote systems
- Use advanced Scheduler concepts to prioritize jobs

ORACLE

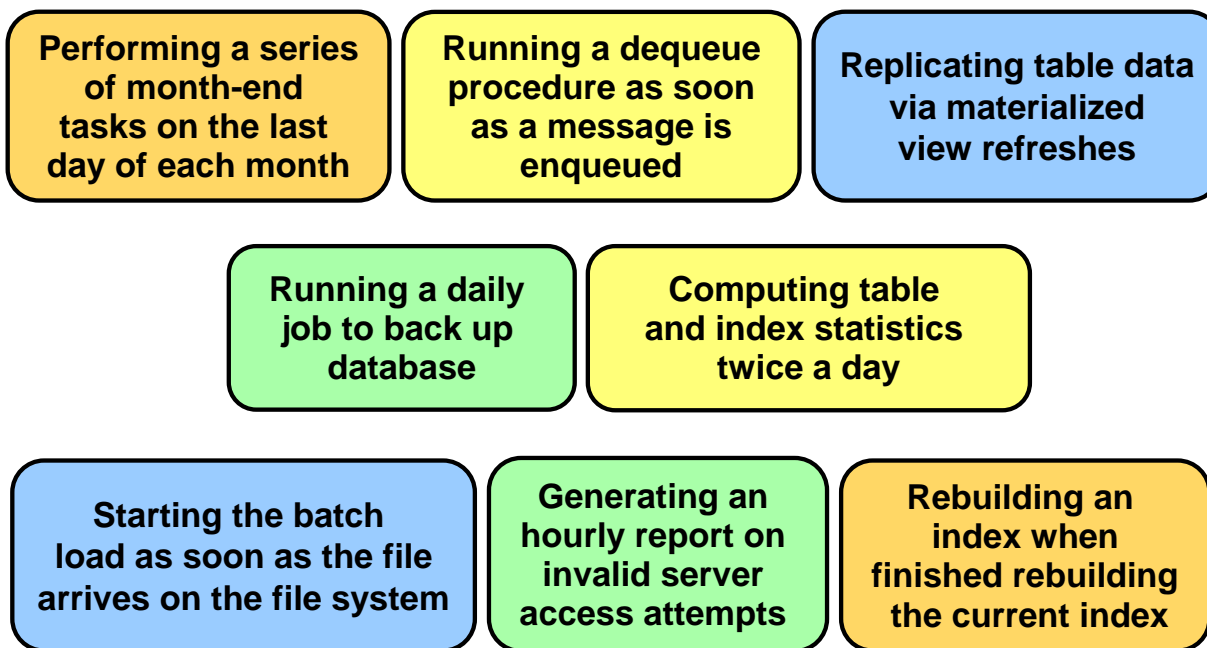
Copyright © 2009, Oracle. All rights reserved.

Objectives

For information about the various Scheduler components and their interaction, see the *Oracle Database Administrator's Guide*.

For detailed information about the DBMS_SCHEDULER package, see the *Oracle Database PL/SQL Packages and Types Reference*.

Simplifying Management Tasks



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Simplifying Management Tasks

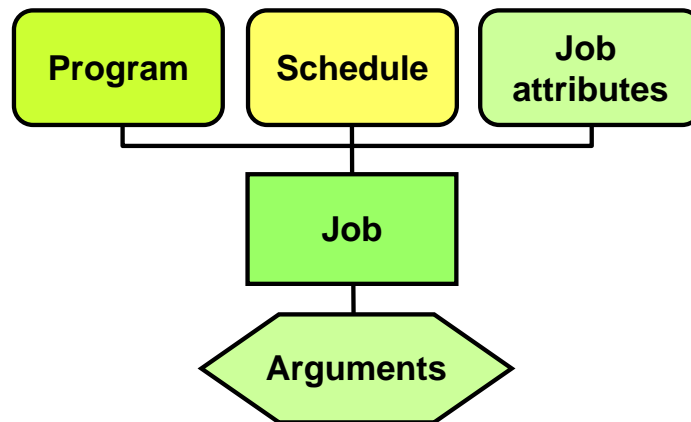
Many tasks in the Oracle environment need job-scheduling capabilities. Routine database maintenance and application logic require jobs to be scheduled and run periodically. Business-to-business (B2B) applications require scheduling for their business events. DBAs need to schedule regular maintenance jobs in specified time windows.

The Oracle database provides advanced scheduling capabilities through the database Scheduler, which is a collection of functions and procedures in the DBMS_SCHEDULER package. The Scheduler can be invoked in any SQL environment, or through Enterprise Manager (EM).

The Scheduler enables database administrators and application developers to control when and where various tasks take place in the database environment. These tasks can be time consuming and complicated; using the Scheduler, you can manage and plan these tasks.

Scheduler jobs can be started based on time or when a specified event occurs, and the Scheduler can raise events when a job's state changes (for example, from RUNNING to COMPLETE). You can also use a named series of programs that are linked together for a combined objective.

Core Components



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Core Components and Key Steps

A job has two mandatory components: “what” action needs to be done and the time or schedule “when” the action occurs. The “what” is expressed in the command region and the job attributes: the `job_type` and the `job_action` parameters. The “when” is expressed in a schedule, which can be based on time or events, or be dependent on the outcome of other jobs.

The Scheduler uses the following basic components:

- A **job** specifies what needs to be executed. It could be as an example a PL/SQL procedure, a native binary executable, a Java application, or a shell script. You can specify the program (what) and schedule (when) as part of the job definition, or you can use an existing program or schedule instead. You can use arguments for a job to customize its run-time behavior.
- A **schedule** specifies when and how many times a job is executed. A schedule can be based on time or an event. You can define a schedule for a job by using a series of dates, an event, or a combination of the two, along with additional specifications to denote repeating intervals. You can store the schedule for a job separately and then use the same schedule for multiple jobs.
- A **program** is a collection of metadata about a particular executable, script, or procedure. An automated job executes some task. Using a program enables you to modify the job task, or the “what,” without modifying the job itself. You can define arguments for a program, enabling users to modify the run-time behavior of the task.

Your Basic Work Flow

To simplify management tasks with the Scheduler:

1. Create a program (enabled or disabled)—optional
 - To reuse this action within multiple jobs
 - To change the schedule for a job without having to re-create the PL/SQL block
2. Create and use a schedule.
3. Create and submit a job.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Your Basic Work Flow

You can perform all steps either in the graphical environment of Enterprise Manager or by using the DBMS_SCHEDULER PL/SQL package via the command line.

1. Creating a Program

Use the CREATE_PROGRAM procedure to create a program. Creating a program is an optional part of using the Scheduler. You can also encode the action to be performed within an anonymous PL/SQL block in the CREATE_JOB procedure. By creating the program separately, you can define the action once, and then reuse this action within multiple jobs. This enables you to change the schedule for a job without having to re-create the PL/SQL block.

A program is created in a disabled state by default (unless the enabled parameter is set to TRUE). A disabled program cannot be executed by a job until it is enabled. You can specify that a program should be created in the enabled state by specifying a value of TRUE for enabled.

2. Creating and Using Schedules

The schedule for a job can be a predefined schedule (created with the CREATE_SCHEDULE procedure) or defined as part of the job creation.

Your Basic Work Flow (continued)

Creating and Using Schedules (continued)

The schedule specifies attributes about when the job is run, such as:

- A start time, which defines when the job is picked for execution and an end time, which specifies the time after which the job is no longer valid and is not scheduled any more
- An expression specifying a repeating interval for the job
- A complex schedule created by combining existing schedules
- A condition or change in state, called an event, that must be met before the job is started

By using a schedule (instead of specifying the execution times for a job within the job definition), you can manage the scheduled execution of multiple jobs without having to update multiple job definitions. If a schedule is modified, each job that uses that schedule automatically uses the new schedule.

3. Creating and Running a Job

A job is a combination of a schedule and a description of what to do, along with any additional arguments that are required by the job. There are many attributes that you can set for a job. Attributes control how the job executes.

Quiz

Select the statements that are true about the Scheduler:

1. Creating a program is a mandatory part of using the Scheduler.
2. When the job action is in a program (rather than directly in the job), you can change the job schedule without having to re-create the PL/SQL block.
3. Creating a job is an optional part of using the Scheduler.
4. Each job must have a schedule. It can be a predefined one or defined as part of the job creation.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 4

Persistent Lightweight Jobs

Persistent lightweight jobs:

- Reduce the overhead and time required to start a job
- Have a small footprint on disk for the job metadata and for storing run-time data
- Are created from a job template (in the command line)

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB (
  job_name          => 'my_lightweight_job2',
  program_name      => 'MY_PROG',
  schedule_name     => 'MY_SCHED',
  job_style         => 'LIGHTWEIGHT');
END;
/
```

Choosing the right job:

- Use regular jobs for maximum flexibility.
- Use persistent lightweight jobs when you need to create a large number of jobs in a very short time.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Lightweight Jobs

A lightweight job:

- Is for customers who need to create hundreds of jobs a second. With regular jobs, each job creates a database object describing the job, modifying several tables, and creating redo in the process. The overhead associated with this type of job need is substantial. In the Oracle Database Scheduler, there is a *persistent lightweight job*. The goal of a lightweight job is to reduce the overhead and time required to start a job. A minimal amount of metadata is created for the job. This reduces the time required and redo created when the job starts.
- Has a small footprint on disk for the job metadata and for storing run-time data. The small footprint on disk also makes load-balancing possible in RAC environments.
- Is always created from a job template. The job template must be a stored procedure or a program. The stored procedure holds all the information needed for the job, including privileges. A few job attributes can be specified: job arguments and schedule.
- Must be created in command line. The JOB_STYLE argument is not available in EM.

In the example, MY_PROG is the job template and the schedule is applied from a named schedule.

Using a Time-Based or Event-Based Schedule

ORACLE Enterprise Manager 11g
Database Control

Database Instance: orcl > Scheduler Jobs > Create Job

Logged in As SYS

General Schedule Options

Schedule Type: Standard

Time Zone: Use Pre-defined Schedule
Standard Using PL/SQL for repeated interval
Use Pre-defined Window

Repeat: Do Not Repeat

Start: Immediate

Date: Jul 13, 2007
(example: Jul 13, 2007)

Time: 6:20:00 AM

Schedule

Time
-Calendaring expression
-Date-time expression

Event

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using a Time-Based or Event-Based Schedule

To specify a time-based schedule for a job, you can specify either a calendaring expression or a date-time expression. When using a calendaring expression, the next start time for a job is calculated using the repeat interval and the start date of the job. When using date-time expressions, the specified expression determines the next time that the job should run. If no repeat interval is specified, the job runs only once on the specified start date.

If a job uses an event-based schedule, the job runs when the event is raised. At a high level, an event can be viewed as a change in state. An event occurs when a Boolean condition changes its state from FALSE to TRUE, or TRUE to FALSE.


The Scheduler uses Oracle Streams Advanced Queuing (AQ) to raise and consume events.

Note: The Scheduler does not guarantee that a job executes on the exact time because the system may be overloaded and thus resources may be unavailable.

Creating a Time-Based Job

Example: Create a job that calls a backup script every night at 11:00, starting tonight.

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB(
    job_name=>'HR.DO_BACKUP',
    job_type => 'EXECUTABLE',
    job_action =>
      '/home/usr/dba/rman/nightly_incr.sh',
    start_date=> SYSDATE,
    repeat_interval=>'FREQ=DAILY;BYHOUR=23',
                      /* next night at 11:00 PM */
    comments => 'Nightly incremental backups');
END;
/
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating a Time-Based Job

Use the `CREATE_JOB` procedure of the `DBMS_SCHEDULER` package to create a job. Jobs are created disabled by default and they become active and scheduled only when they are explicitly enabled. All job names are of the form: [schema .]name.

You should use `SYSTIMESTAMP` and specify a time zone so that when the time changes because of daylight saving time, your job adjusts its execution time automatically.

By default, a job is created in the current schema. You can create a job in another schema by specifying the name of the schema, as shown in the example in the slide. The job owner is the user in whose schema the job is created, whereas the job creator is the user who created the job. Jobs are executed with the privileges of the job owner. The national language support (NLS) environment of the job when it runs is the same as that present at the time the job was created. The `job_type` parameter indicates the type of task to be performed by the job. The possible values are:

- **PLSQL_BLOCK:** An anonymous PL/SQL block
- **STORED_PROCEDURE:** A named PL/SQL, Java, or external procedure
- **EXECUTABLE:** A command that can be executed from the operating system (OS) command line

Creating a Time-Based Job (continued)

The `job_action` parameter can be the name of the procedure to run, the name of a script or operating system command, or an anonymous PL/SQL code block, depending on the value of the `job_type` parameter.

In the example in the slide, `job_type` is specified as `EXECUTABLE` and `job_action` is the full OS-dependent path of the desired external executable plus optionally any command-line arguments.

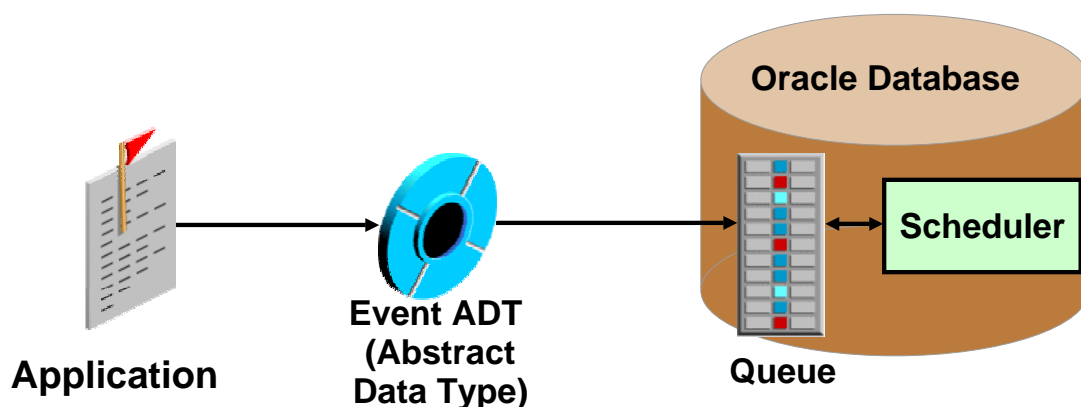
An external job is a job that runs outside the database. All external jobs run as a low-privileged guest user, as has been determined by the database administrator while configuring external job support. Because the executable is run as a low-privileged guest account, you should verify that it has access to necessary files and resources. Most, but not all, platforms support external jobs. For platforms that do not support external jobs, creating or setting the attribute of a job or a program to type `EXECUTABLE` returns an error.

Refer to your Oracle database platform-specific documentation for more information about configuring the environment to run external programs with the Scheduler.

Creating an Event-Based Schedule

To create an event-based job, you must set:

- A queue specification (where your application enqueues messages to start a job)
- An event condition (same syntax as an Oracle Streams AQ rule condition) that if `TRUE` starts the job



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating an Event-Based Schedule

Jobs can be triggered based on events. An application can notify the Scheduler to start a job by enqueueing a message onto an Oracle Streams queue. A job started in this way is referred to as an event-based job. To create an event-based job, you must set the following two additional attributes with the `CREATE_JOB` procedure:

- **queue_spec:** A queue specification that includes the name of the queue where your application enqueues messages to raise job start events, or in the case of a secure queue, the `<queue_name>`, `<agent_name>` pair
- **event_condition:** A conditional expression based on message properties that must evaluate to `TRUE` for the message to start the job. You can include user data properties in the expression, provided that the message payload is a user-defined object type, and that you prefix object attributes in the expression with `tab.user_data`.

You can either specify `queue_spec` and `event_condition` as in-line job attributes, or create an event-based schedule with these attributes and then create a job that references this schedule.

Creating Event-Based Schedules with Enterprise Manager

The screenshot shows the 'Schedule' configuration page in Oracle Enterprise Manager. It is divided into two main sections: 'Schedule' and 'Event Parameters'. In the 'Schedule' section, the 'Time Zone' is set to 'America/New_York' and the 'Schedule Type' is set to 'Event'. The 'Event Parameters' section contains three fields: 'Queue Name' is 'SYS.ALERT_QUE' with a 'Change Queue' button; 'Agent Name' is 'ADMIN_AGNT1'; and 'Condition' is 'tab.user_data.event_type = "DISK_FULL" '.

Schedule	
Time Zone	America/New_York
Schedule Type	Event
Event Parameters	
★ Queue Name	SYS.ALERT_QUE Change Queue
★ Agent Name	ADMIN_AGNT1
★ Condition	tab.user_data.event_type = "DISK_FULL" '

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Event-Based Schedules with Enterprise Manager

The Create Schedule page enables you to choose between a standard, time-based schedule and an event-based schedule. If you choose an event-based schedule, then the interface changes and you can specify the queue name, agent name, and event condition, in addition to the other schedule attributes.

Note: The Scheduler runs the event-based job for each occurrence of an event that matches `event_condition`. However, events that occur while the job is already running are ignored; the event gets consumed, but does not trigger another run of the job.

References

- See the *Oracle Streams Advanced Queuing User's Guide and Reference* for information about how to create queues and enqueue messages.
- For more information about Oracle Streams AQ rules and event conditions, see the `DBMS_AQADM.ADD_SUBSCRIBER` procedure in the *Oracle Database PL/SQL Packages and Types Reference 11* manual.

Creating an Event-Based Job

Example: Create a job that runs if a batch load data file arrives on the file system before 9:00 AM.

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB(
    job_name=>'ADMIN.PERFORM_DATA_LOAD',
    job_type => 'EXECUTABLE',
    job_action => '/loaddir/start_my_load.sh',
    start_date => SYSTIMESTAMP,
    event_condition => 'tab.user_data.object_owner =
event_condition => 'tab.user_data.object_owner =
  'HR' and tab.user_data.object_name = 'DATA.TXT'
  and tab.user_data.event_type = 'FILE_ARRIVAL'
  and tab.user_data.event_timestamp < 9 ',
    queue_spec => 'HR.LOAD_JOB_EVENT_Q');
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating an Event-Based Job

To specify event information as job attributes, you use an alternate syntax of `CREATE_JOB` that includes the `queue_spec` and `event_condition` attributes. The job can include event information in-line as job attributes or can specify event information by pointing to an event schedule. The example shown in the slide uses an in-line, event-based schedule.

The example in the slide shows a job that is started when a file arrives on the operating system, as long as the file arrives before 9:00 AM. Assume that the message payload is an object with four attributes named `object_owner`, `object_name`, `event_type`, and `event_timestamp`.

The example uses a user-defined event. Therefore, before this job can be started, when the file arrives on the file system, a program or procedure must enqueue the event object type with the proper information into the specified event queue. The `HR.LOAD_JOB_EVENT_Q` queue must be of the same type as the event object type used for notifying the Scheduler of an event occurrence. That is, the `HR.LOAD_JOB_EVENT_Q` queue must be a typed queue where the type has four attributes named `object_owner`, `object_name`, `event_type`, and `event_timestamp`.

For more information about how to create queues and enqueue messages, refer to the *Oracle Streams Advanced Queuing User's Guide and Reference* documentation.

Event-Based Scheduling

Event types:

- User- or application-generated events
- Scheduler-generated events

Events raised by Scheduler jobs:

- | | |
|-----------------|-----------------------|
| • JOB_STARTED | • JOB_SCH_LIM_REACHED |
| • JOB_SUCCEEDED | • JOB_DISABLED |
| • JOB_FAILED | • JOB_CHAIN_STALLED |
| • JOB_BROKEN | • JOB_ALL_EVENTS |
| • JOB_COMPLETED | • JOB_RUN_COMPLETED |
| • JOB_STOPPED | • JOB_OVER_MAX_DUR |

Example of raising an event:

```
DBMS_SCHEDULER.SET_ATTRIBUTE( 'hr.do_backup',  
    'raise_events', DBMS_SCHEDULER.JOB_FAILED);
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Event-Based Scheduling

You can create a job that directly references an event as the means to start the job, instead of assigning a schedule to the job. There are two types of events:

- **User- or application-generated events:** An application can raise an event to be consumed by the Scheduler. The Scheduler reacts to the event by starting a job. An example of such events: a running job completes; a file arrives on the file system; an account within the database is locked; and the inventory reaches a low threshold.
- **Scheduler-generated events:** The Scheduler can raise an event to indicate state changes that occur within the Scheduler itself. For example, the Scheduler can raise an event when a job starts, when a job completes, when a job exceeds its allotted run time, and so on. The consumer of the event is an application that performs some action in response to the event.

You can configure a job so that the Scheduler raises an event when the job's state changes. You do this by setting the `raise_events` job attribute. By default, a job does not raise any state change events until you alter the `raise_events` attribute for a job. To alter this attribute, you must first create the job by using the `CREATE_JOB` procedure and then use the `SET_ATTRIBUTE` procedure to modify the attribute's default value. The example shows that the `hr.do_backup` job is altered, so that it raises an event if the job fails.

Event-Based Scheduling (continued)

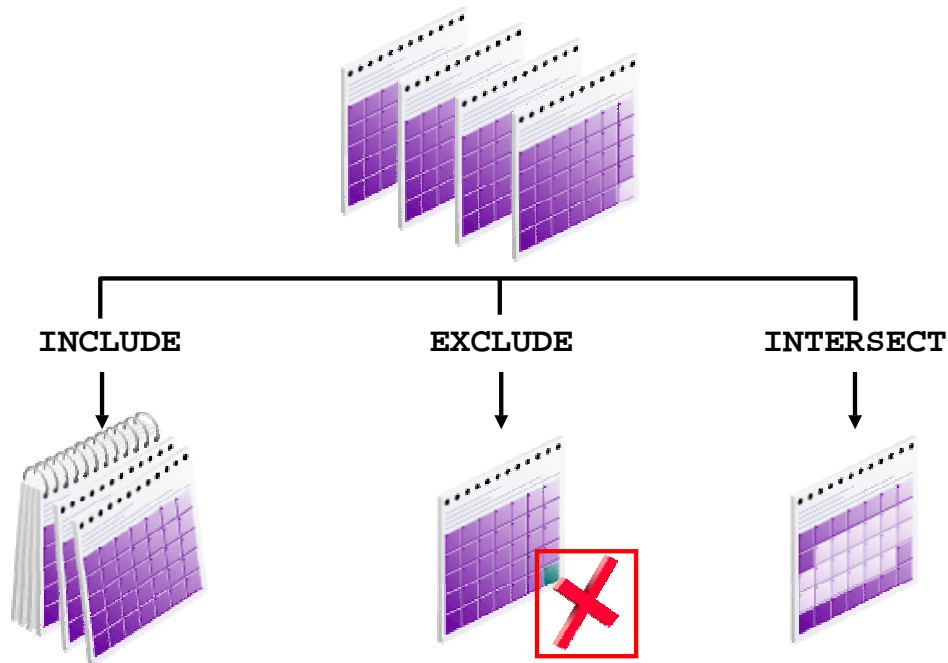
After you enable job state change events for a job, the Scheduler raises these events by enqueueing messages onto the default event queue `SYS.SCHEDULER$_EVENT_QUEUE`.

The default Scheduler event queue is a secure queue. Depending on your application, you may have to configure the queue to enable certain users to perform operations on it. See the *Oracle Streams Concepts and Administration* documentation for information about secure queues.

The default Scheduler event queue is intended primarily for Scheduler-generated events. Oracle does not recommend the use of this queue for user applications, or user-defined events.

Event Type	Description
JOB_STARTED	The job is started.
JOB_SUCCEEDED	The job is successfully completed.
JOB_FAILED	The job failed, either by raising an error or by abnormally terminating.
JOB_BROKEN	The job is disabled and changed to the BROKEN state, because it exceeded the number of failures defined by the MAX_FAILURES job attribute.
JOB_COMPLETED	The job is completed, because it reached the values set by the MAX_RUNS or END_DATE job attributes.
JOB_STOPPED	The job is stopped by a call to the STOP_JOB procedure.
JOB_SCH_LIM_REACHED	The job's schedule limit is reached. The job is not started, because the delay in starting the job exceeded the value of the SCHEDULE_LIMIT job attribute.
JOB_DISABLED	The job is disabled by the scheduler or by a call to the SET_ATTRIBUTE procedure.
JOB_CHAIN_STALLED	A job running a chain is put into the CHAIN_STALLED state. A running chain becomes stalled if there are no steps running or scheduled to run and the chain EVALUATION_INTERVAL is set to NULL. The chain waits for manual intervention.
JOB_ALL_EVENTS	JOB_ALL_EVENTS is not an event, but a constant, that provides an easy way for you to enable all events.
JOB_OVER_MAX_DUR	The job has run over the maximum time it was set to be allowed to run.
JOB_RUN_COMPLETED	A job run is completed. It either failed, succeeded, or is stopped.

Creating Complex Schedules



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Complex Schedules

A schedule is an object in the database. When you create schedules, they are automatically saved. You can use combinations of schedules to create more complex schedules. By combining schedules, you can add specific dates to or exclude specific dates from a calendaring expression.

You can use the following options when defining the repeat interval for a schedule:

- **INCLUDE:** Adds a list of dates to the calendaring expression results
- **EXCLUDE:** Removes a list of dates from the calendaring expression results
- **INTERSECT:** Uses only the dates that are common to two or more schedules

When creating schedules to be used in combinations, you can code the list of dates by including hard-coded dates of the form [YYYY]MMDD or by including named schedules created with the `CREATE_SCHEDULE` procedure. For example, you can specify a list of dates by using the following values for the repeat interval of a schedule:

```
0115,0315,0325,0615,quarter_end_dates,1215
```

This string represents the dates January 15, March 15, March 25, June 15, December 15, and the list of dates specified by the `QUARTER_END_DATES` schedule.

If you do not specify the optional year component for hard-coded dates in your schedule, the dates are included for every year.

Quiz

Select the statements that are true about persistent lightweight jobs:

1. Persistent lightweight jobs have a small footprint on disk for the job metadata and also for storing run-time data.
2. Use persistent lightweight jobs for maximum flexibility.
3. Persistent lightweight jobs are created from a job template.
4. Persistent lightweight jobs can be created in Enterprise Manager and via command line.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 3

Using Email Notification

- Email notifications for change of job state
- Triggered by job state events
- Multiple notifications, multiple recipients
- *_SCHEDULER_NOTIFICATIONS views

Using Scheduler Email Notification:

1. Specify the address of the SMTP server you will use to send email messages:

```
DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE  
( 'email_server', 'host[:port]' );
```

2. Optionally, set a default sender email address:

```
DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE  
( 'email_sender', 'valid email address' );
```

3. Add email notifications for a specified job. *(continued)*

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Email Notification

The job email notification feature enables you to add email notifications to existing jobs so that events of interest that happen to the job are emailed to specified email addresses. For each job, you can add notifications for different events. You can send the email notification to more than one recipient.

To enable the email notification feature, you must:

1. Set the email_server Scheduler attribute.
2. Optionally, you can use the email_sender Scheduler attribute to specify a default sender email address for the email notifications.
3. After creating a job, execute the DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION procedure to add one or more notifications for the job.

The data dictionary supports email notifications with the *_SCHEDULER_NOTIFICATIONS views.

Adding and Removing Email Notifications

```
DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION (
  job_name          IN VARCHAR2,
  recipients        IN VARCHAR2,
  sender            IN VARCHAR2 DEFAULT NULL,
  subject           IN VARCHAR2
    DEFAULT dbms_scheduler.default_notification_subject,
  body              IN VARCHAR2
    DEFAULT dbms_scheduler.default_notification_body,
  events            IN VARCHAR2
    DEFAULT 'JOB_FAILED,JOB_BROKEN,JOB_SCH_LIM_REACHED,
             JOB_CHAIN_STALLED,JOB_OVER_MAX_DUR',
  filter_condition  IN VARCHAR2 DEFAULT NULL);
```

Comma-separated list of email addresses

Mandatory comma-separated list

```
DBMS_SCHEDULER.REMOVE_JOB_EMAIL_NOTIFICATION (
  job_name          IN VARCHAR2,
  recipients        IN VARCHAR2 DEFAULT NULL,
  events            IN VARCHAR2 DEFAULT NULL);
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Adding and Removing Email Notifications

Add one or more job email notifications with the `DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION` procedure. Email messages will be sent to the specified recipient addresses whenever any of the listed events are generated by the job. The job is automatically modified to raise these events. If a filter condition is specified, only events that match the specification in `FILTER_CONDITION` will generate an email message.

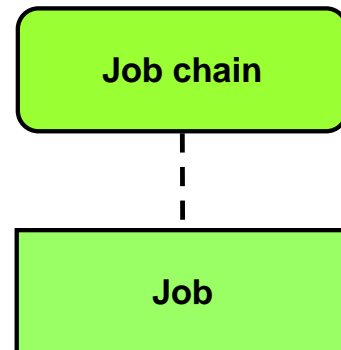
This procedure will fail if the `EMAIL_SERVER` scheduler attribute is not set or if the specified job does not exist. The user calling this procedure must be the owner of the job, have the `CREATE ANY JOB` system privilege, or have been granted the `ALTER` privilege for the job.

- The **subject** of notification emails can contain the following variables for which values will be substituted: `%job_owner%`, `%job_name%`, `%event_type%`, `%event_timestamp%`, `%log_id%`, `%error_code%`, `%error_message%`, `%run_count%`, `%failure_count%`, `%retry_count%`, `%job_subname%`, `%job_class_name%`.
- The notification email message **body** can contain any of the variables that are valid in the `SUBJECT`.
- The comma-separated list of **events** cannot be `NULL`. Refer to the list of events for the `RAISE_EVENTS` attribute of `JOBS` for valid events.
- If **filter_condition** is `NULL` (the default), all occurrences of the specified events will be emailed to all specified recipient addresses.

Remove one or more email notifications for a specified job with the `DBMS_SCHEDULER.REMOVE_JOB_EMAIL_NOTIFICATION` procedure.

Creating Job Chains

1. Create a chain object.
2. Define chain steps.
3. Define chain rules.
4. Starting the chain:
 - Enable the chain.
 - Create a job that points to the chain.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Job Chains

A chain is a named series of programs that are linked together for a combined objective. This is known as “dependency scheduling.” An example of a chain may be the following:

Run program A and then program B, but only run program C if programs A and B complete successfully, otherwise run program D.

Each position within a chain of interdependent programs is referred to as a step. Typically, after an initial set of chain steps has started, the execution of successive steps depends on the completion of one or more previous steps. To create and use a chain, you complete the following steps in order. All procedures mentioned are part of the DBMS_SCHEDULER package, unless noted otherwise.

1. **Create a chain** by using the CREATE_CHAIN procedure. The chain name can be optionally qualified with a schema name (for example, *myschema.myname*).
2. **Define** (one or more) **chain steps**. Defining a step gives it a name and specifies what happens during the step. Each step can point to one of the following:
 - A program
 - Another chain (a nested chain)
 - An event

You define a step that points to a program or nested chain by calling the DEFINE_CHAIN_STEP procedure.

Creating Job Chains (continued)

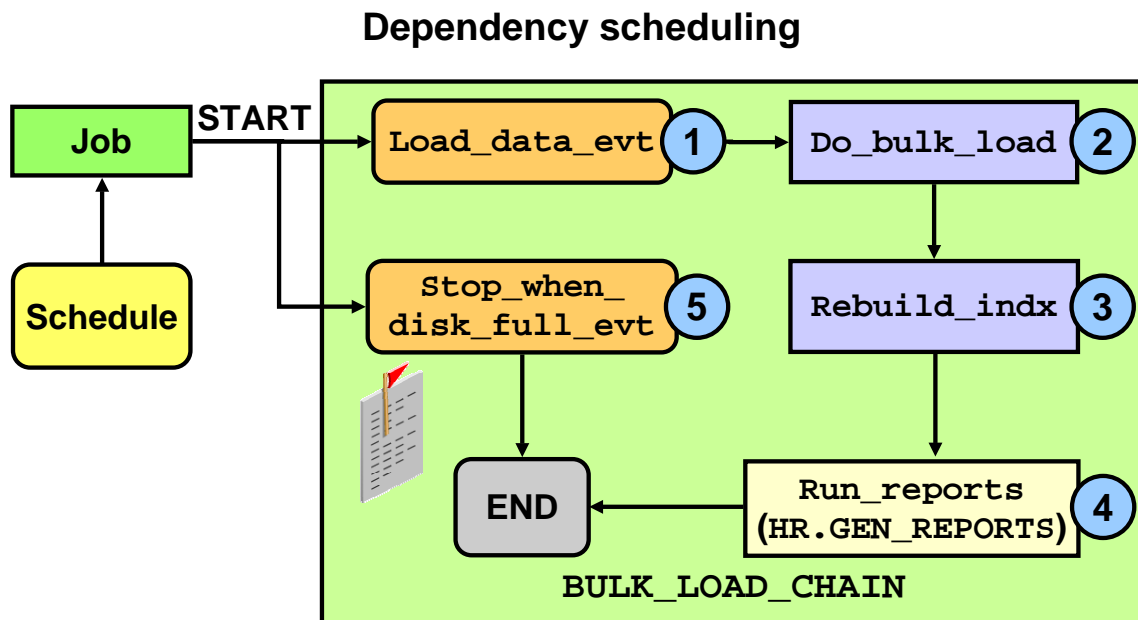
To define a step that waits for an event to occur, you use the `DEFINE_CHAIN_EVENT_STEP` procedure. Procedure arguments can point to an event schedule or can include an in-line queue specification and event condition. A step that points to an event waits until the specified event is raised. If the event occurs, the step completes successfully.

3. After creating the chain object, you **define chain rules**. Chain rules define when steps run, and define dependencies between steps. Each rule has a *condition* and an *action*:
 - If the condition evaluates to `TRUE`, the action is performed. The condition can contain any syntax that is valid in a `SQL WHERE` clause. Conditions are usually based on the outcome of one or more previous steps. For example, you may want one step to run if the two previous steps succeeded, and another to run if either of the two previous steps failed.
 - The action specifies what is to be done as a result of the rule being triggered. A typical action is to run a specified step. Possible actions include starting or stopping a step. You can also choose to end the execution of the job chain, returning either a value or a step name and error code.

All rules added to a chain work together to define the overall behavior of the chain. When the job starts and at the end of each step, all rules are evaluated to see what action or actions occur next. You add a rule to a chain with the `DEFINE_CHAIN_RULE` procedure. You call this procedure once for each rule that you want to add to the chain.

4. **Starting the chain** involves two actions:
 - Enable a chain with the `ENABLE` procedure. (A chain is always created disabled, so you can add steps and rules to the chain before it is executed by any job.) Enabling an already enabled chain does not return an error.
 - To run a chain, you must create a job of type `'CHAIN'`. The job action must refer to the chain name. You can use either event-based or time-based schedules for this job.

Example of a Chain



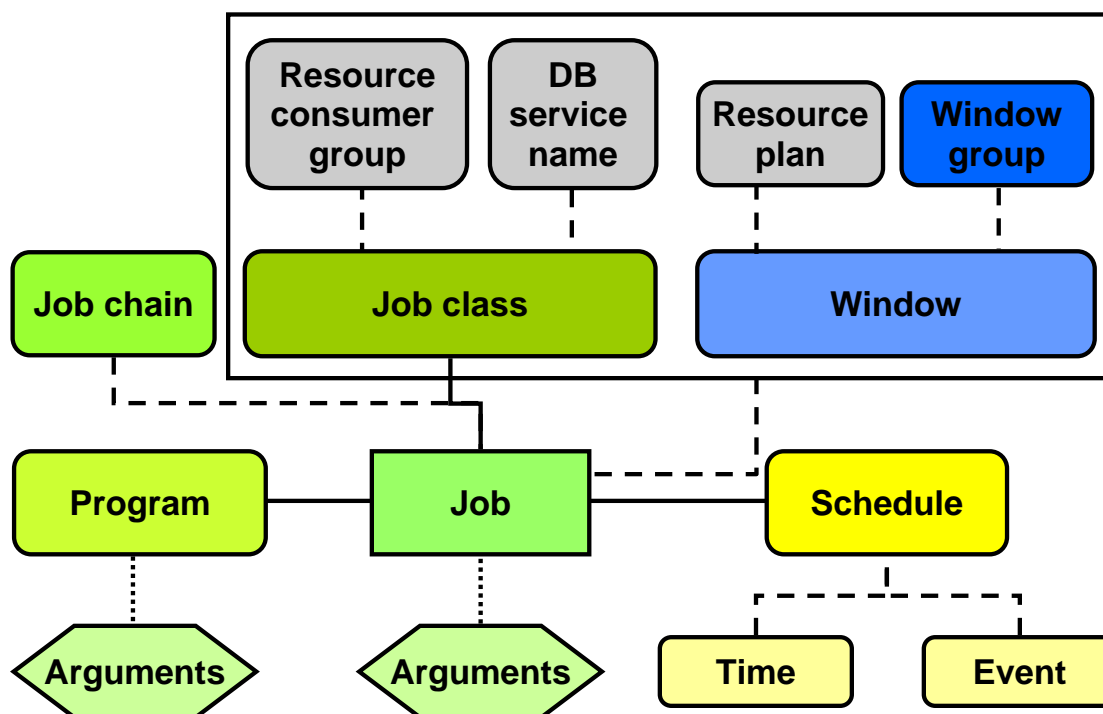
ORACLE

Copyright © 2009, Oracle. All rights reserved.

Example of a Chain

As an example of a chain, consider all the tasks and conditions that occur during a bulk data load. First, you must have data to load. Then load the data, observing the file system to make sure that you do not run out of space during the load. After the data load completes, you need to rebuild the indexes defined on the updated tables. Then you run reports against the newly loaded data. The slide shows an example of dependency scheduling.

Advanced Scheduler Concepts



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Advanced Scheduler Concepts

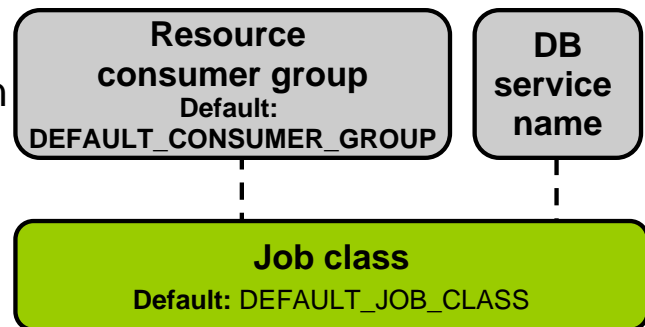
Using advanced Scheduler features, you can exercise more control over various aspects of scheduling, such as job windows and prioritizing jobs. These advanced features are summarized below, and are discussed in detail in the following slides.

- A **window** is represented by an interval of time with a well-defined beginning and end, and is used to activate different resource plans at different times. This allows you to change resource allocation during a time period such as time of day or time of the sales year.
- A **window group** represents a list of windows, and allows for easier management of windows. You can use a window or window group as the schedule for a job to ensure that the job runs only when a window and its associated resource plans are active.
- A **job class** defines a category of jobs that share common resource usage requirements and other characteristics. A job class groups jobs into larger entities.
- A **resource consumer group** associated with the job class determines the resources that are allocated to the jobs in the job class.
- A **resource plan** enables users to prioritize resources (most notably CPU) among resource consumer groups.

Note: The gray objects are not Scheduler objects.

Job Classes

- Assign the same set of attribute values to member jobs
- Are created by the `CREATE_JOB_CLASS` procedure
- Specify jobs in a job class (with the `SET_ATTRIBUTE` procedure)
- Belong to the `SYS` schema
- Set resource allocation for member jobs
- Set the service attribute to a desired database service name
- Group jobs for prioritization



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Job Classes

Job classes are policies for their participating member jobs. Each job class specifies a set of attributes, such as the logging level. When you assign a job to a job class, the job inherits those attributes. For example, you can specify the same policy for purging log entries for all payroll jobs.

- You can use the `CREATE_JOB_CLASS` procedure to create a job class. A class always belongs to the `sys` schema. To create a class, you must have the `MANAGE_SCHEDULER` privilege. There is a default job class named `DEFAULT_JOB_CLASS` that is created with the database
- After a job class has been created, you can specify jobs as members of this job class when you create the jobs, or after the jobs are created, by using the `SET_ATTRIBUTE` procedure of the `DBMS_SCHEDULER` package. If a job is not associated with a job class, the job belongs to this default job class.
- Set the service attribute of a job class to a desired database service name. This determines the instances in a Real Application Clusters environment that run the member jobs, and optionally the system resources that are assigned to the member jobs.

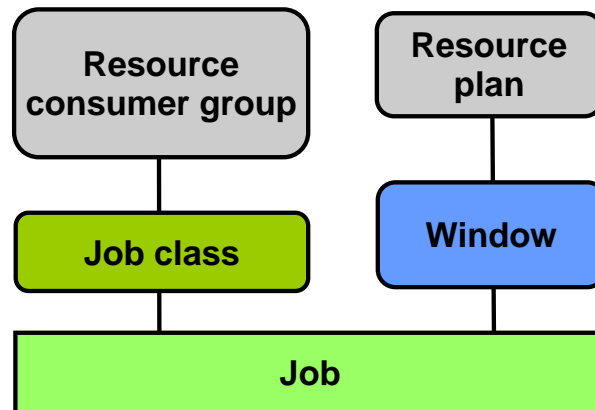
Job Classes (continued)

- Set resource allocation for member jobs. Job classes provide the link between the Database Resource Manager and the Scheduler because each job class can specify a resource consumer group as an attribute. Member jobs then belong to the specified consumer group and are assigned resources according to settings in the current resource plan. Alternatively, you can leave the `resource_consumer_group` attribute as NULL and set the service attribute of a job class to a desired database service name. That service can in turn be mapped to a resource consumer group. If both the `resource_consumer_group` and service attributes are set, and the designated service maps to a resource consumer group, the resource consumer group named in the `resource_consumer_group` attribute takes precedence. If a resource consumer group is not specified when a job class is created, the job class maps to the `DEFAULT_CONSUMER_GROUP` resource consumer group. Jobs in the default job class or in a job class associated with the default resource consumer group may not be allocated enough resources to complete their tasks when the Resource Manager is enabled.
- Group jobs for prioritization. Within the same job class, you can assign priority values of 1–5 to individual jobs so that if two jobs in the class are scheduled to start at the same time, the one with the higher priority takes precedence. This ensures that you do not have a less important job preventing the timely completion of a more important one. If two jobs have the same assigned priority value, the job with the earlier start date takes precedence. If no priority is assigned to a job, its priority defaults to 3.

Windows

Scheduler windows:

- Can start jobs or change resource allocation among jobs for various time periods
- One active at a time
- Created with the `CREATE_WINDOW` procedure



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Windows

The priority of jobs can change over a period of time. For example, you might want to allocate a high percentage of the database resources to data warehouse loading jobs at night and allocate a higher percentage of the resources during the day to the application jobs. To accomplish this, you can change the database resource plan by using a Scheduler window.

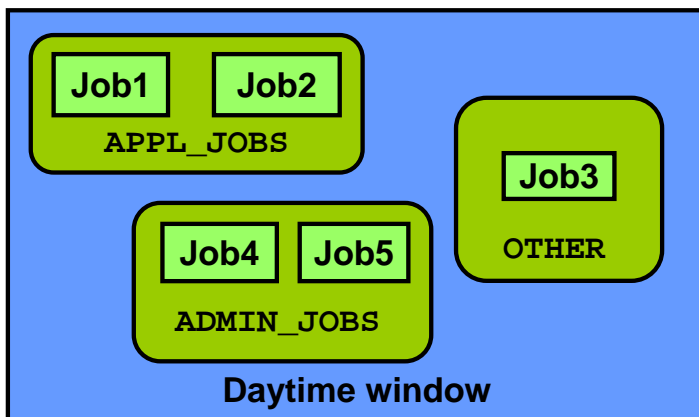
- Scheduler windows can automatically start jobs or change the resource allocation among jobs during various time periods of the day, week, and so on. A window is represented by an interval of time with a well-defined beginning and end, such as “from 12:00 AM to 6:00 AM.”
- Only one window can be in effect at any given time.
- You can create a window with the `CREATE_WINDOW` procedure.

Scheduler windows work with job classes to control resource allocation. Each window specifies the resource plan to activate when the window opens (becomes active), and each job class specifies a resource consumer group or specifies a database service, which can map to a consumer group. A job that runs within a window, therefore, has resources allocated to it according to the consumer group of its job class and the resource plan of the window (as shown in the graphic in this slide).

Prioritizing Jobs Within a Window

Prioritizing jobs:

- At the class level (via resource plans)
- At the job level (with the job priority attribute)
- Not guaranteed for jobs in different job classes



Job	Priority
Job1	1
Job2	2
Job3	3
Job4	5
Job5	2

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Prioritizing Jobs Within a Window

When creating multiple jobs in a database, you need a way to align the job processing with your business requirements and specify which jobs have the highest priority. For a particular window, you may have several classes of jobs running, each with its own priority.

There are two levels at which jobs can be prioritized: at the class level and at the job level.

- The first prioritization is at the class level, using resource plans. Prioritization among jobs of different classes is done purely on a class resource allocation basis.
- The second prioritization is within the class, with the job priority attribute of the job.

Prioritization levels are relevant only when two jobs within the same class are supposed to start at the same time. The job with the higher priority starts first.

Prioritization is not guaranteed for jobs in different job classes. For example, a high-priority job in the APPL_JOBS job class might not get started before a low-priority job in the ADMIN_JOBS job class, even if they share the same schedule. If the APPL_JOBS job class has a lower level of resource available, the high-priority job in that class has to wait for resources to become available, even if there are resources available to lower-priority jobs in a different job class.

Creating a Job Array

1. Declare variables of types `sys.job` and `sys.job_array`:

```
DECLARE
  newjob sys.job;
  newjobarr sys.job_array;
```

2. Initialize the job array:

```
BEGIN
  newjobarr := SYS.JOB_ARRAY();
```

3. Size the job array to hold the number of jobs needed:

```
newjobarr.EXTEND(100);
```

(... continued)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating a Job Array

A more efficient way to create a set of jobs is the use of a job array. This also applies to lightweight jobs. In the example in the slide, 100 job specifications are created in a job array and submitted to the job queue in a single transaction. Notice that for a lightweight job there is a very limited amount of information needed. In the example, the `start_time` parameter defaults to `NULL`, so the job is scheduled to start immediately.

1. Declare the variable to hold a job definition and a job array variable.
2. Initialize the job array using the `SYS.JOB_ARRAY` constructor. This creates a place for one job in the array.
3. Set the size of the array to the number of expected jobs.
4. Create each job and place it in the array. In the slide example, the only difference is the name of the job. The `start_time` variable of the job is omitted and it defaults to `NULL`, indicating that the job will run immediately.
5. Use the `CREATE_JOBS` procedure to submit all the jobs in the array as one transaction.

Note: If the array is very small, the performance will not be significantly better than submitting a single job.

Creating a Job Array

4. Place jobs in the job array:

```
FOR i IN 1..100 LOOP
    newjob := SYS.JOB(job_name => 'LWTJK' || to_char(i),
                     job_style => 'LIGHTWEIGHT',
                     job_template => 'MY_PROG',
                     enabled => TRUE );
    newjobarr(i) := newjob;
END LOOP;
```

5. Submit the job array as one transaction:

```
DBMS_SCHEDULER.CREATE_JOBS(newjobarr,
                           'TRANSACTIONAL');
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating a Job Array (continued)

The full code of this example is:

```
DECLARE
    newjob sys.job;
    newjobarr sys.job_array;
BEGIN
    -- Create an array of JOB object types
    newjobarr := sys.job_array();
    -- Allocate sufficient space in the array
    newjobarr.extend(100);
    -- Add definitions for jobs
    FOR i IN 1..100 LOOP
        -- Create a JOB object type
        newjob := sys.job(job_name => 'LWTJK' || to_char(i),
                         job_style => 'LIGHTWEIGHT',
                         job_template => 'PROG_1',
                         enabled => TRUE );

        -- Add job to the array
        newjobarr(i) := newjob;
    END LOOP;
    -- Call CREATE_JOBS to create jobs in one transaction
    DBMS_SCHEDULER.CREATE_JOBS(newjobarr, 'TRANSACTIONAL');
END;
/
```

Quiz

Select the statements that are true about the advanced Scheduler concepts and functionality:

1. Lightweight jobs can be created with a job array.
2. Prioritizing jobs at the class level (via resource plans) and at the job level (with the job priority attribute) are mutually exclusive.
3. Scheduler windows work with job classes to control resource allocation.
4. Job chains are used to implement “dependency scheduling.”

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1, 3, 4

Creating a File Watcher and an Event-Based Job

Perform the following tasks:

1. Create a Scheduler credential object and grant `EXECUTE`.
2. Create a file watcher and grant `EXECUTE`.
3. Create a Scheduler program object with a metadata argument that references the event message.
4. Create an event-based job that references the file watcher. (Optionally, enable the job to run for each instance of the file arrival event.)
5. Enable the file watcher, the program, and the job.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating a File Watcher and an Event-Based Job

Perform the following tasks to create a file watcher and create the event-based job that starts when the designated file arrives:

1. Create a Scheduler credential object (a credential) with which to authenticate with the host operating system for access to the file and grant `EXECUTE` on the credential to the schema that owns the event-based job that the file watcher will start.
2. Create a file watcher and grant `EXECUTE` on the file watcher to any schema that owns an event-based job that references the file watcher.
3. Create a Scheduler program object with a metadata argument that references the event message.
 - Define the metadata argument using the `EVENT_MESSAGE` attribute.
 - Create the stored procedure with an argument of the `SYS.SCHEDULER_FILEWATCHER_RESULT` type that the program invokes. The stored procedure must have an argument of the `SYS.SCHEDULER_FILEWATCHER_RESULT` type, which is the data type of the event message. The position of that argument must match the position of the defined metadata argument. The procedure can access attributes of this abstract data type to learn about the arrived file.

Creating a File Watcher and an Event-Based Job (continued)

4. Create an event-based job that references the file watcher. You can use the `DBMS_SCHEDULER.SET_ATTRIBUTE` procedure to enable the job to run for each instance of the file arrival event, even if the job is already processing a previous event. Set the `PARALLEL_INSTANCES` attribute to `TRUE`.

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE( ' ', 'PARALLEL_INSTANCES', TRUE );
END;
```

This enables the job to run as a lightweight job so that multiple instances of the job can be started quickly. If `PARALLEL_INSTANCES` is set to the default value of `FALSE`, file watcher events that occur while the event-based job is already processing another will be discarded.

5. Enable the file watcher, the program, and the job.

Enabling File Arrival Events from Remote Systems

Perform the following tasks to enable the raising of file arrival events at remote systems:

1. Set up the database to run remote external jobs.
2. Install, configure, register, and start the Scheduler agent on the first remote system.
3. Repeat step 2 for each additional remote system.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Enabling File Arrival Events from Remote Systems

To receive file arrival events from a remote system, you must install the Scheduler agent on that system, and you must register the agent with the database. The remote system does not need an Oracle Database instance to generate file arrival events.

Refer to the *Oracle Database Administrator's Guide 11g Release 2* for detailed information.

Scheduling Remote Database Jobs

- Create a job that runs stored procedures and anonymous PL/SQL blocks on another database instance on the same host or a remote host.
- The target database can be any release of Oracle Database.
- `DBMS_SCHEDULER.CREATE_DATABASE_DESTINATION` and `DBMS_SCHEDULER.CREATE_CREDENTIAL` can be used for remote database jobs.
- Jobs with job types of `PLSQL_BLOCK` and `STORED_PROCEDURE` can be the subject of `SET_ATTRIBUTE` calls for the `DESTINATION` and `CREDENTIAL` attributes.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Scheduling Remote Database Jobs

You can now create a job that runs stored procedures and anonymous PL/SQL blocks on another database instance on the same host or a remote host. The target database can be any release of Oracle Database.

There are no new procedures to support remote database jobs, rather there have been changes to existing `DBMS_SCHEDULER` procedures to support this functionality. Additional information is provided over the next few pages.

Creating Remote Database Jobs

Perform the following tasks to create a remote job:

1. Set up the originating database for remote jobs.
2. Create the job by using `DBMS_SCHEDULER.CREATE_JOB`.
3. Create a credential by using `DBMS_SCHEDULER.CREATE_CREDENTIAL`.
4. Set the job `CREDENTIAL_NAME` attribute by using `DBMS_SCHEDULER.SET_ATTRIBUTE`.
5. Set the job `DESTINATION` attribute by using `DBMS_SCHEDULER.SET_ATTRIBUTE`.
6. Enable the job by using `DBMS_SCHEDULER.ENABLE`.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Remote Database Jobs

You can perform the tasks listed in the slide to create a remote database job.

To set up the originating database for remote jobs, perform the following steps:

1. Verify that XML DB is installed.
2. Enable HTTP connections to the database.

```
BEGIN
DBMS_XDB.SETHTTPPORT(port);
END;
```
3. Execute the `prvtrsch.plb` script.
4. Set a registration password for the Scheduler agents.

```
BEGIN
DBMS_SCHEDULER.SET_AGENT_REGISTRATION_PASS('password');
END;
```

Refer to the *Oracle Database Administrator's Guide 11g Release 2* for a detailed example.

Scheduling Multiple Destination Jobs

- This enables you to specify several targets on which your jobs should execute.
- It provides the ability to monitor and control the jobs from the database on which they were created.
- While running, a multiple-destination job is viewed as a collection of jobs, which are near-identical copies of each other.
- All jobs will execute based on the time zone that is specified in the start date of the job or will use the time zone of the source database.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Scheduling Multiple Destination Jobs

The multiple destination jobs feature enables you to specify multiple targets on which the jobs should execute. You have the ability to monitor and control the jobs from the database at which you created them. The following capabilities are included:

- Specifying several databases or machines on which a job must execute
- Modifying a job scheduled on multiple targets as a single entity
- Stopping or dropping jobs running on one or more remote targets
- Finding out the status of job instances on all of a job's targets

Note that in the initial release of this feature, all destinations will run based on the time zone that is specified in the start date of the job or will default to the time zone of the source database.

Viewing Scheduler Meta Data

Major Scheduler management views, displaying:

- *_SCHEDULER_JOBS: All jobs, enabled and disabled
- *_SCHEDULER_SCHEDULES: All schedules
- *_SCHEDULER_PROGRAMS: All programs
- *_SCHEDULER_RUNNING_JOBS: Active job states
- *_SCHEDULER_JOB_LOG: All job state changes
- *_SCHEDULER_JOB_RUN_DETAILS: All completed job runs

```
SELECT job_name, status, error#, run_duration
FROM USER_SCHEDULER_JOB_RUN_DETAILS;
```

JOB_NAME	STATUS	ERROR#	RUN_DURATION
-----	-----	-----	-----
GATHER_STATS_JOB	SUCCESS	0	+000 00:08:20
PART_EXCHANGE_JOB	FAILURE	6576	+000 00:00:00

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Viewing Scheduler Meta Data

The job table is a container for all the jobs, with one table per database. The job table stores information for all jobs such as the owner name or the level of logging. You can find this information in the *_SCHEDULER_JOBS views.

Jobs are database objects, and can, therefore, accumulate and take up too much space. To avoid this, job objects are automatically dropped by default after completion. This behavior is controlled by the auto_drop job attribute.

There are many views available to the DBA and privileged users that provide essential operating information regarding the scheduler, jobs, schedules, windows, and so on. These views include:

- *_SCHEDULER_PROGRAM_ARGS: Shows all arguments defined for all programs as well as the default values if they exist
- *_SCHEDULER_JOBS: Shows all jobs, enabled as well as disabled
- *_SCHEDULER_JOB_RUN_DETAILS show all completed (failed or successful) job runs. It has a row for each job instance. Each row contains information about the job execution for that instance. The ERROR# is the number of the first error encountered
- *_SCHEDULER_GLOBAL_ATTRIBUTE: Shows the current values of Scheduler attributes
- *_SCHEDULER_JOB_ARGS: Shows all set argument values for all jobs
- *_SCHEDULER_JOB_CLASSES: Shows all job classes

Scheduler Data Dictionary Views (continued)

For job chains:

- *_SCHEDULER_RUNNING_CHAINS: Shows all active chains
- *_SCHEDULER_CHAIN_STEPS: Shows all steps for all chains
- *_SCHEDULER_CHAINS: Shows all chains
- *_SCHEDULER_CHAIN_RULES: Shows all rules for all chains

For windows and other advanced objects:

- *_SCHEDULER_WINDOWS show all windows.
- *_SCHEDULER_WINDOW_GROUPS show all window groups.
- *_SCHEDULER_WINDOWGROUP_MEMBERS show the members of all window groups, one row for each group member.
- *_SCHEDULER_JOB_LOG show all state changes made to jobs.
- *_SCHEDULER_CREDENTIALS displays a list of credentials in the database with obfuscated passwords.
- *_SCHEDULER_JOB_ROLES show all jobs by database role.

Lightweight jobs are visible through the same views as regular jobs are:

- *_SCHEDULER_JOBS: Shows all jobs, including the JOB_STYLE= 'LIGHTWEIGHT'
- *_SCHEDULER_JOB_ARGS: Shows all set argument values also for lightweight jobs
- Because lightweight jobs are not database objects, they are not visible through the *_OBJECTS views.

Starting with the Oracle Database 11gR2:

- *_SCHEDULER_NOTIFICATIONS shows which email notifications have been set.
- *_SCHEDULER_FILE_WATCHERS shows file watcher configuration information.

The following views display information about multiple destination jobs:

- *_SCHEDULER_DESTS: Shows all the destinations on which remote jobs can be scheduled. The views contain both the external destinations (for remote external jobs) as well as the database destinations for remote database jobs.
- *_SCHEDULER_EXTERNAL_DESTS: Shows all the agents that have registered with the database and can be used as a destination for remote external jobs.
- *_SCHEDULER_DB_DESTS: Shows all the databases on which you can schedule remote database jobs.
- *_SCHEDULER_GROUPS: Shows the groups in your schema or all groups in the database.
- *_SCHEDULER_GROUP_MEMBERS: Shows the group members in your schema or all group members in the database.
- *_SCHEDULER_JOB_DESTS: Shows the state of a job at a remote database.

Note: In the views listed above, the asterisk at the beginning of a view name can be replaced with DBA, ALL, or USER.

Quiz

Select the statements that are true about the Oracle Scheduler:

1. Creating remote database jobs is a manual task, requiring the use of OS-specific commands.
2. A Scheduler credential is an object with which to authenticate with the host operating system for file access.
3. You can specify several targets on which your jobs should execute and monitor them from the database on which they were created.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 3

Summary

In this lesson, you should have learned how to:

- Simplify management tasks by using the Scheduler
- Create a job, program, and schedule
- Monitor job execution
- Use a time-based or event-based schedule for executing Scheduler jobs
- Describe the use of windows, window groups, job classes, and consumer groups
- Use email notification
- Use job chains to perform a series of related tasks
- Describe Scheduler jobs on remote systems
- Use advanced Scheduler concepts to prioritize jobs

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 17 Overview: Automating Tasks with the Scheduler

This practice covers the following topics:

- Creating a job that runs a program outside the database
- Creating a program and a schedule
- Creating a job that uses a program and a schedule
- Create a lightweight job
- Monitoring job runs

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Note

This practice uses both Enterprise Manager Database Control and SQL*Plus.

18

Managing Space

ORACLE

Copyright © 2009, Oracle. 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

ORACLE

Copyright © 2009, Oracle. 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)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Space Management: Overview

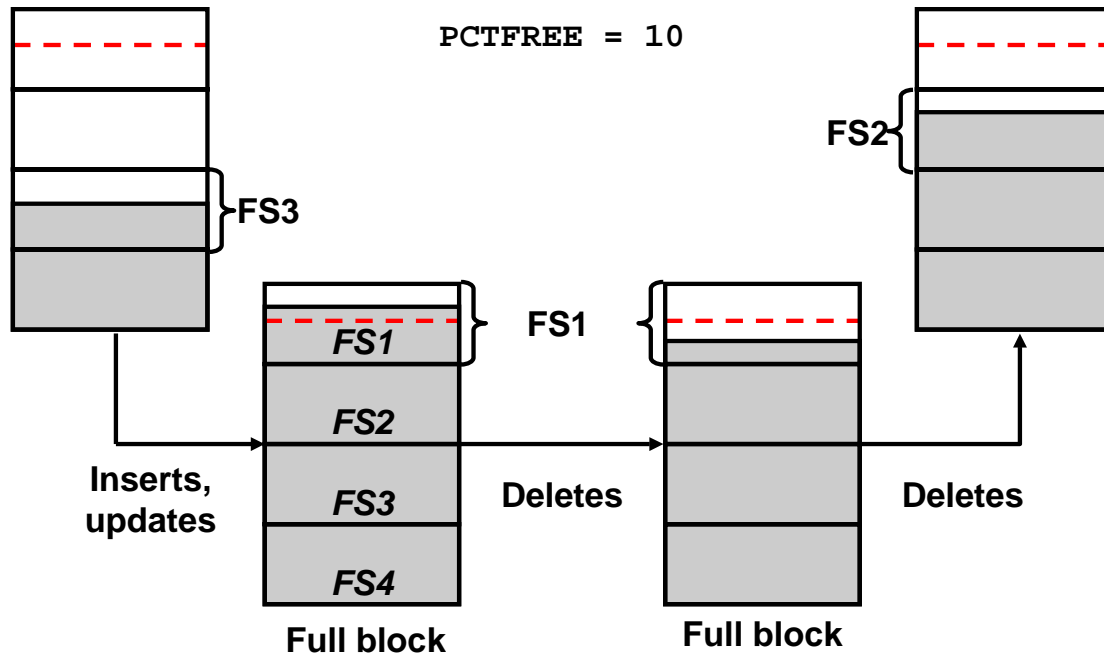
With Oracle Managed Files (OMF), you can specify operations in terms of database objects rather than file names. The Oracle database server can manage free space within a tablespace with bitmaps. This is known as a “locally managed” tablespace. In addition, free space within segments located in locally 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).

(The topics are covered in this and the following lesson.)

Block Space Management



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Block Space Management

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.

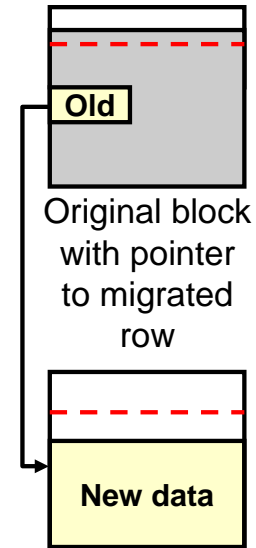
In the slide example, the block on the left is an FS3 block because it has between 50% and 75% free space. After some insert and update statements, $PCTFREE$ is reached (the dashed line) and it is no longer possible to insert new rows in that block. The block is now considered as a “full” or FS1 block. 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 © 2009, Oracle. All rights reserved.

Row Chaining and Migrating

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.

Row Chaining and Migrating (continued)

The Oracle database 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.

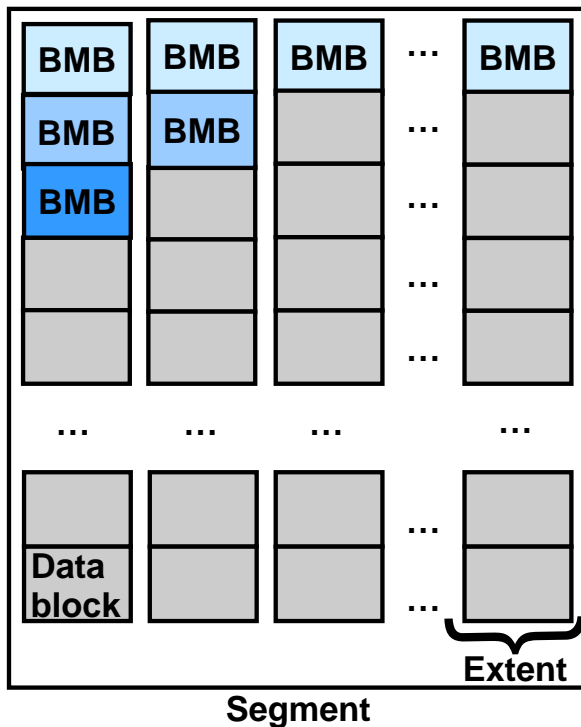
1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Free Space Management Within Segments



- Tracked by bitmaps in segments

Benefits:

- More flexible space utilization
- Run-time adjustment
- Multiple process search of BMBs

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Free Space Management

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 the 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 run-time adjustment to variations in concurrent access
- Better multi-instance behavior in terms of performance or space utilization

Therefore, less work for you, the DBA.

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.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Types of Segments

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.

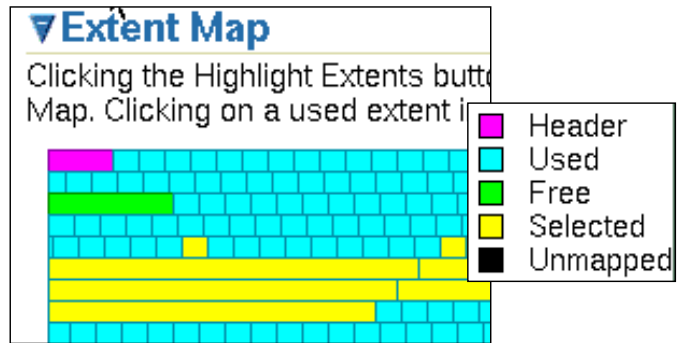
Undo segment: Oracle Database maintains information to reverse changes made to the database. This information consists of records of the actions of transactions, collectively known as undo. Undo is stored in undo segments in an undo tablespace.

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 © 2009, Oracle. All rights reserved.

Allocating Extents

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.

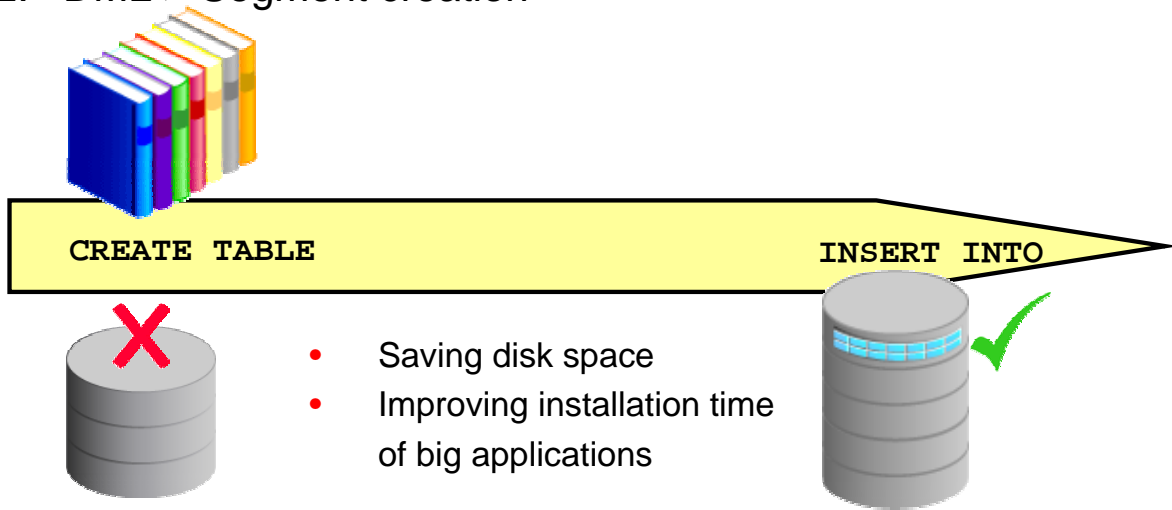
To view the extent map in Enterprise Manager, choose Server > Tablespaces > View Tablespace > Show Tablespace Contents.

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.

Allocating Space

New space allocation method:

- `DEFERRED_SEGMENT_CREATION = TRUE` (default)
1. Table creation > Data dictionary operation
 2. DML > Segment creation



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Allocating Space

Oracle Database 11gR2 includes a new space allocation method. When you create a nonpartitioned heap table, the 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 new 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 new 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*.

Creating Tables Without Segments

```
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, 'aaaaaaa');
1 row created.

SQL> SELECT segment_name FROM user_segments;
SEGMENT_NAME
-----
SEG_TEST
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating Tables Without Segments

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.

Another addition to the data dictionary is the `SYS.SEG$` table that 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 in Oracle Database 11gR2)

```
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.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Controlling Deferred Segment Creation

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 11gR2.
 - `SEGMENT CREATION IMMEDIATE`: If specified, segments are materialized during table creation. This is the default behavior in Oracle databases prior to Oracle Database 11gR2.

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:

- Only for nonpartitioned tables and indexes
- Not for IOTs, clustered tables, or other special tables
- Not for tables in dictionary-managed tablespaces

Note: 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.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Restrictions and Exceptions

- In Oracle Database 11gR2, deferred segment creation is restricted to nonpartitioned tables and nonpartitioned indexes. IOTs and other special tables are not supported.
- 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.
- Segment creation on demand is not supported for 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.

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';
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Additional Automatic Functionality

Additional enhancements in Oracle Database 11g Release 2 (unrelated to the deferred segment creation) are implemented to save space: all UNUSABLE indexes and index partitions are created without a segment. This functionality is completely transparent for you. It is enabled by default with the COMPATIBILITY initialization parameter set to 11.2.0.0.

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 for Oracle Database 11g Release2?

1. Deferred segment creation is always enabled. You cannot control it.
2. You can control the deferred segment creation with the `SEGMENT CREATION` clause of the `CREATE TABLE` command.
3. Segment creation on demand is available for all types of tables, including those owned by the `SYS` user.
4. Segment creation on demand is available for nonpartitioned tables.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 4

Table Compression: Overview

Reducing storage costs by compressing all data:

- Basic compression for direct-path insert operations: 10x
- OLTP compression for all DML operations: 2–4x

```
{ COMPRESS [ BASIC | FOR { OLTP } ] | NOCOMPRESS }
```

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

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Table Compression: Overview

Oracle Database supports three methods of table compression:

- Basic table compression
- OLTP table compression
- Hybrid columnar compression (with Exadata)

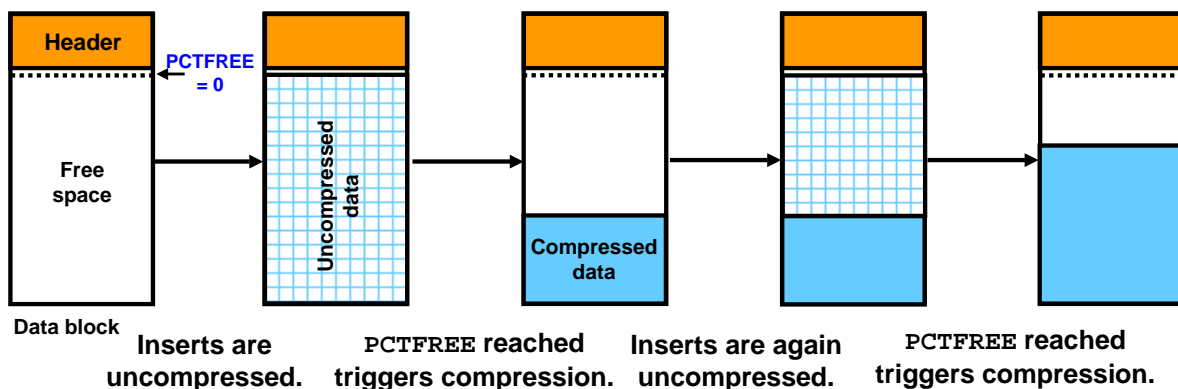
Oracle Corporation recommends to compress all data to reduce storage costs. The Oracle database can use table compression to eliminate duplicate values in a data block. For tables with highly redundant data, compression saves disk space and reduces memory use in the database buffer cache.

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 compresses data at the time of performing bulk load using operations such as direct loads or `CREATE TABLE AS SELECT`.
- With `COMPRESS FOR OLTP`, the Oracle database 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
- Replaces deprecated `COMPRESS FOR DIRECT_LOAD OPERATIONS`
- Maximizes contiguous free space in blocks



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Compression for Direct-Path Insert Operations

With `COMPRESS` or `COMPRESS BASIC`, you enable basic table compression.

- The Oracle database 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.

Compression for Direct-Path Insert Operations (continued)

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 `COMPRESS FOR OLTP` or `NOCOMPRESS` use the PCTFREE default value of 10 to maximize compress while still allowing for some future DML changes to the data, unless you override this default explicitly.

OLTP Compression for DML Operations

- Is enabled with
`CREATE TABLE ... COMPRESS FOR OLTP ...;`
- Is recommended for active OLTP environments
- Replaces deprecated `COMPRESS FOR ALL OPERATIONS`

	Y		Y		Y
G		Y		G	
	G		Y	Y	G

Uncompressed
block

G	Y				
	Y		Y		Y
G		Y		G	
	G		Y	Y	G

OLTP compression with symbol table at
the beginning of the block

ORACLE

Copyright © 2009, Oracle. All rights reserved.

OLTP Compression for DML Operations

With `COMPRESS FOR OLTP`, you enable OLTP table 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`. This syntax has been deprecated.

With OLTP 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 OLTP 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 OLTP 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 OLTP compressed

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying Table Compression

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:

- `COMPRESS FOR OLTP` 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.
- Logical standby, Streams, and LogMiner are not supported on hybrid columnar compressed tables.

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 EM)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the Compression Advisor

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 EM, 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_uncmp,
    rowcnt_cmp, rowcnt_uncmp, comptype);
DBMS_OUTPUT.PUT_LINE('Blk count compressed = ' || blkcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Blk count uncompressed = ' ||
    blkcnt_uncmp);
DBMS_OUTPUT.PUT_LINE('Row count per block compressed = ' ||
    rowcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed = ' ||
    rowcnt_uncmp);
DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype);
DBMS_OUTPUT.PUT_LINE('Compression ratio =
    ' || blkcnt_uncmp/blkcnt_cmp || ' to 1');
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the DBMS_COMPRESSION Package

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 procedure returns the compression type for a given row.

For more details, see the *Oracle Database PL/SQL Packages and Types Reference*.

Compressing Table Data

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

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Compressing Table Data

Oracle Database supports three methods of table compression:

- Basic table compression
- OLTP table compression
- Hybrid columnar compression (with Exadata)

Proactive Tablespace Monitoring

The screenshot displays the Oracle Enterprise Manager 11g interface for monitoring and configuring tablespaces. At the top, the 'Tablespaces' page is shown with a search bar and a table of tablespaces. A red arrow points from the 'Edit' button in the table to the 'Edit Tablespace: EXAMPLE' configuration page on the right. Another red arrow points from the 'Thresholds' tab in the configuration page to the 'Tablespace Full Metric Thresholds' section on the left.

Tablespaces Table:

Select	Name	Allocated Size(MB)	Space Used(MB)	Allocated Space Used(%)	Allocated Free Space(MB)	Status	Datafiles	Type	Extent Management	Segment Management
<input checked="" type="radio"/>	EXAMPLE	100.0	77.4	<div><div></div></div>	22.6	✓	1	PERMANENT	LOCAL	AUTO

Edit Tablespace: EXAMPLE

Actions: Add Datafile | Go | Show SQL | Revert

Tabs: General | Storage | Thresholds

Extent Allocation
Allocation Type: Automatic

Segment Space Management
Type: Automatic

Options
Compress: ☐ Yes ☒ No
Enable data segment compression to reduce disk usage. This option is especially useful in environments such as data warehouses, where the amount of insert and update operations is small.

Tablespace Full Metric Thresholds
Monitor the fullness of the tablespace using either of the metrics below.

Space Used (%)
A warning or critical alert will be generated if the percentage of space used exceeds the corresponding threshold.

☒ Use Database Default Thresholds (Modify)
Warning (%) 85
Critical (%) 97

☐ Specify Thresholds
Warning (%)
Critical (%)

☐ Disable Thresholds

Free Space (MB)
A warning or critical alert will be generated if the remaining free space falls below the corresponding threshold. This metric is especially useful for large tablespaces.

☒ Use Database Default Thresholds (Modify)
Warning (MB) Not Defined
Critical (MB) Not Defined

☐ Specify Thresholds
Warning (MB)
Critical (MB)

☐ Disable Thresholds

ORACLE
Copyright © 2009, Oracle. All rights reserved.

Proactive Tablespace Monitoring

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, select Server from the Database Home page, and then select Tablespaces. Select the tablespace of your choice and click the Edit button.

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 © 2009, Oracle. All rights reserved.

Thresholds and Resolving Space Problems

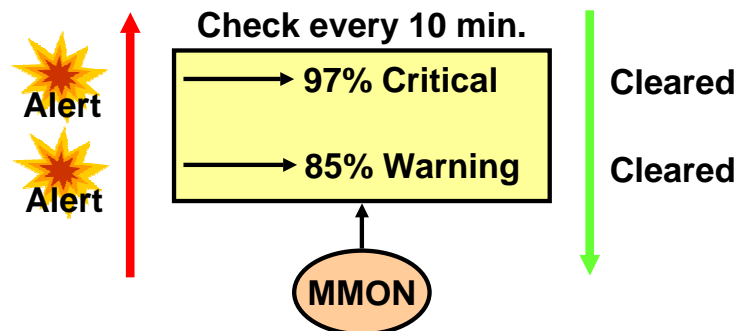
Tablespace thresholds are defined either as full or as available space in the tablespace. Critical and warning thresholds are the two thresholds that apply to a tablespace. The `DBMS_SERVER_ALERT` package contains procedures to set and get the threshold values. When the tablespace limits are reached, 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 autoextendable
- Freeing up space on disks that contain any autoextendable 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.
- Autoextensible files: Threshold is based on the maximum file size.

ORACLE

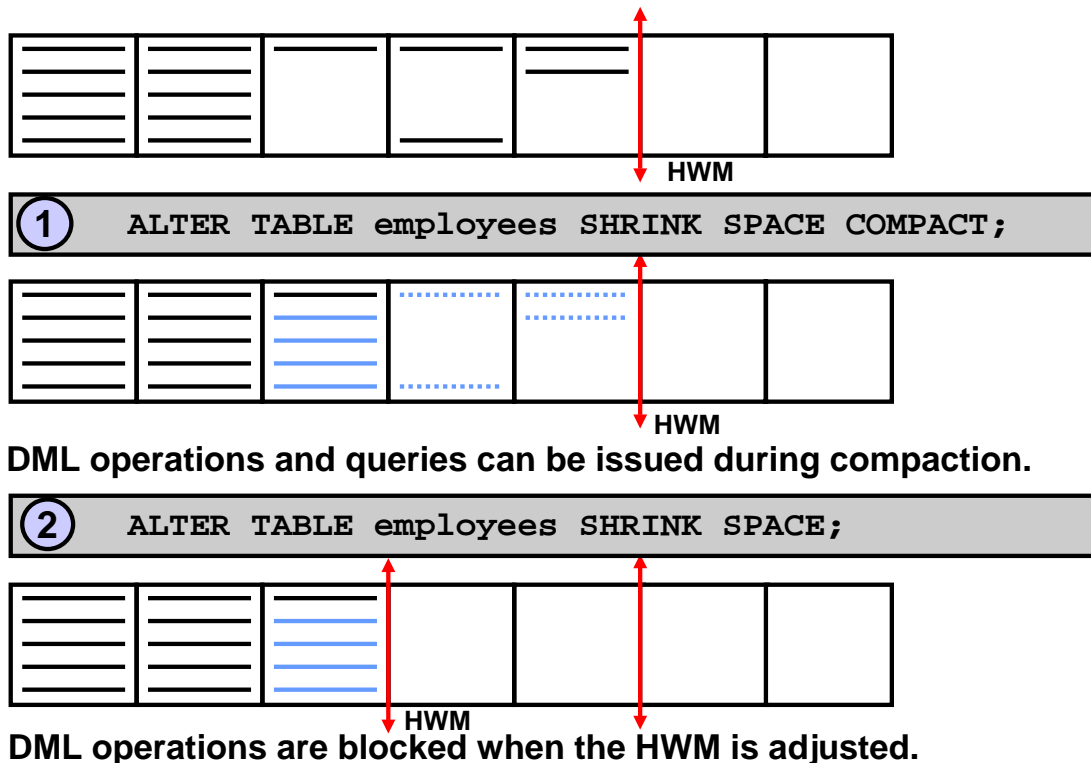
Copyright © 2009, Oracle. All rights reserved.

Monitoring Tablespace Space Usage

The database tracks space utilization while performing regular space management activities. This information is aggregated every 10 minutes 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 autoextensible files, the thresholds are computed according to the maximum file size you specified, or the maximum OS file size.

Shrinking Segments



ORACLE

Copyright © 2009, Oracle. All rights reserved.

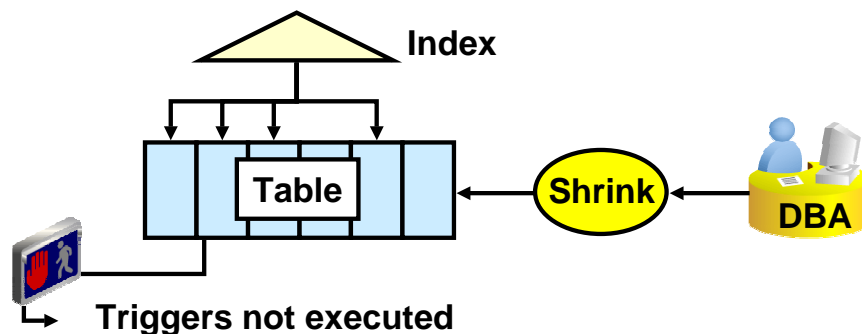
Shrinking Segments

The diagram in the slide describes the two phases of a table shrink operation. The first phase does the compaction. 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 © 2009, Oracle. All rights reserved.

Results of Shrink Operation

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, any 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

ORACLE

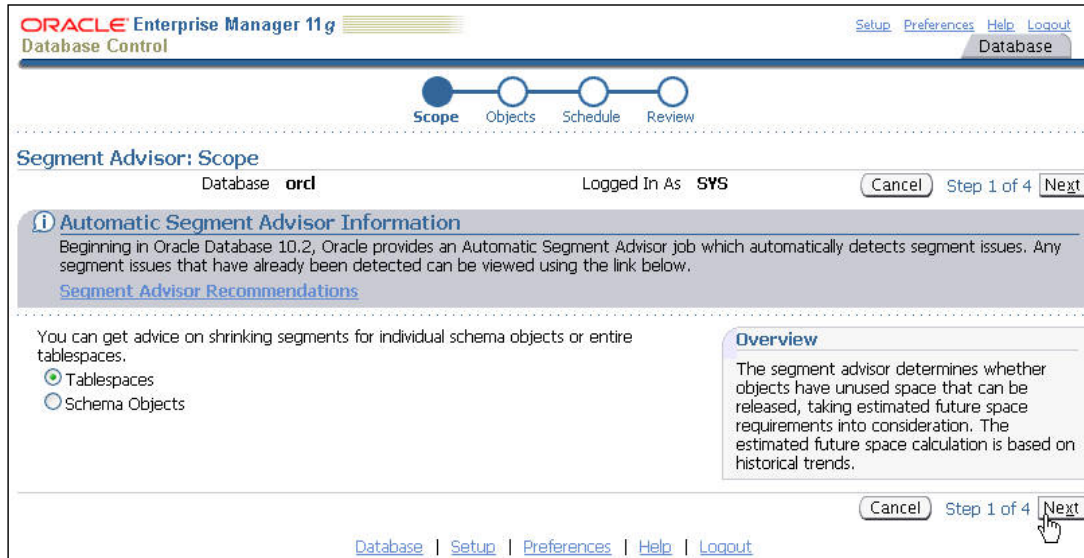
Copyright © 2009, Oracle. All rights reserved.

Space Reclamation with ASSM

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.

Segment Advisor: Overview



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Segment Advisor: Overview

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.

The EM Database Control Console is the interface to the Segment Advisor. You can access the Segment Advisor from several places within EM:

- Advisor Central page
- Tablespaces page
- Schema object pages

The Database Control Console 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.

Segment Advisor

Tablespaces: Add

Database: orcl Logged In As: SYS

Search:

Available Tablespaces

Select	Name	Type	Extent Management	Segment Space Management	Size (MB)	Used (MB)	Used (%)
<input type="checkbox"/>	SYSTEM	PERMANENT	LOCAL	MANUAL	710.00	701.56	98.81
<input type="checkbox"/>	TEMP	TEMPORARY	LOCAL	MANUAL	54.00	53.00	98.15
<input type="checkbox"/>	SYSAUX	PERMANENT	LOCAL	AUTO	796.69	757.88	95.13
<input checked="" type="checkbox"/>	EXAMPLE	PERMANENT	LOCAL	AUTO			
<input type="checkbox"/>	UNDOTBS1	UNDO	LOCAL	MANUAL			
<input type="checkbox"/>	USERS	PERMANENT	LOCAL	AUTO			

Segment Advisor: Tablespaces

Database: orcl Logged In As: SYS

Step 2 of 4

Name	Type	Extent Management	Segment Space Management	Size (MB)	Used (MB)	Used (%)
EXAMPLE	PERMANENT	LOCAL	AUTO	100.00	77.38	77.38

Options: [Show Advanced Options](#)

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Segment Advisor

From the Server page, select Tablespaces in the Storage section. On the Tablespaces page, select the tablespace on which you want to perform the shrink analysis, and then select Run Segment Advisor in the Actions drop-down list. Click Go to open the Segment Advisor initial page. You must choose “comprehensive” or “limited” analysis mode. In comprehensive mode, the analysis is longer because the advisor is sampling the segments to identify the right targets.

Keep clicking Continue to answer the various questions of the advisor. You end up on the Segment Advisor: Review page, where you can review the details of your analysis. The Segment Advisor analysis is run as a scheduled job, so you can review the scheduled task from the Advisor Central page. When completed, you can review the advisor’s recommendations.

Note: In the Segment Advisor, you can specify the duration of the analysis. This enables you to limit the time the advisor takes to produce recommendations. Generally speaking, a longer analysis period produces more comprehensive results. The results are stored in the AWR and can be viewed later. Use the “Number of days to retain” option to instruct the Oracle database server how long these results should be preserved before being purged from the AWR.

Implementing Recommendations

Advisor Central

[Advisors](#) [Checkers](#)

Page Refreshed Jul 12, 2007 7:18:54 AM EDT [Refresh](#)

Advisors

[ADDM](#) [Automatic Undo Management](#) [Data Recovery Advisor](#)
[Memory Advisors](#) [MTTR Advisor](#) [Segment Advisor](#)
[SQL Advisors](#) [SQL Performance Analyzer](#)

Advisor Tasks [Change Default Parameters](#)

Search

Select an advisory type and optionally enter a task name to filter the data that is displayed in your results set.

Advisory Type: Task Name: Advisor Runs: Status: [Go](#)

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Results

[View Result](#) [Delete](#) Actions: [Re-schedule](#) [Go](#) [Previous](#) 1-25 of 45 [Next 20](#)

Select	Advisory Type	Name	Description	User	Status	Start Time	Duration (seconds)	Expires In (days)
<input checked="" type="checkbox"/>	Segment Advisor	SEGMENTADV_5412893	Get shrink advice based on object growth trend	SYS	COMPLETED	Jul 12, 2007 7:13:44 AM	134	30

Select	Schema	Segment	Recommendation	Reclaimable Space (MB)	(MB)	Type
<input checked="" type="checkbox"/>	SYS	EMPLOYEES1	Shrink	alter table "SYS"."EMPLOYEES1"	shrink space	
<input checked="" type="checkbox"/>	SYS	EMPLOYEES2	Shrink	alter table "SYS"."EMPLOYEES2"	shrink space	
<input checked="" type="checkbox"/>	SYS	EMPLOYEES3	Shrink	alter table "SYS"."EMPLOYEES3"	shrink space	

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Implementing Recommendations

After the Segment Advisor completes its job, you can view the recommendation details and implement them directly.

Note: Before shrinking a heap-organized table, you must enable row movement on that table. You can do this with Database Control from the Options tab on the Edit Table page.

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

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Automatic Segment Advisor

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.

Manual Segment Shrink Using EM

Oracle Enterprise Manager 11g Database Control

Database Instance: orcl > Logged in As SYS (Recycle Bin)

Tables

Object Type: Table

Search

Enter a schema name and an object name to filter the data that is displayed in your results set.

Schema: Object Name:

Go

By default, the search returns all uppercase matches beginning with the string you entered. To run an exact or case-sensitive match, double quote the search string. You can use the wildcard symbol (%) in a double quoted string.

Selection Mode: ☒ Single

Edit View Delete With Options Actions Shrink Segment Go

Select	Schema	Table Name	Tablespace	Partitioned	Rows	Last Analyzed
<input type="radio"/>	HR	COUNTRIES	EXAMPLE	NO	25 Jun 22, 2007 4:16:21 PM	
<input type="radio"/>	HR	DEPARTMENTS	EXAMPLE	NO	27 Jun 22, 2007 4:16:22 PM	
<input type="radio"/>	HR	DEPARTMENTS2	EXAMPLE	NO	30 Jul 3, 2007 10:00:17 PM	
<input checked="" type="radio"/>	HR	EMPLOYEES	EXAMPLE	NO	107 Jun 22, 2007 4:16:22 PM	

Oracle Enterprise Manager 11g Database Control

Database Instance: orcl > Tables > Shrink Segment: HR.EMPLOYEES

Segment Name: HR.EMPLOYEES Object Type: Table

Show SQL Cancel Continue

The shrink operation compacts fragmented space and, optionally, frees the space. The shrink operation will take some time and will be scheduled as a job.

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 phase, any cursors referencing this segment may be invalidated and queries on the segment could be affected.

☐ Compact Segments
Compacting will compact segment data without releasing the recovered space. After compacting the data, the recovered space can be quickly released by running Compact Segments and Release Space.

Segment Selection

☒ Shrink HR.EMPLOYEES Only
☐ Shrink HR.EMPLOYEES and All Dependent Segments

Dependent Segments

Schema	Segment Name	Type	Tablespace
HR	EMPLOYEES	TABLE	EXAMPLE
HR	EMP_EMAIL_UK	INDEX	EXAMPLE
HR	EMP_EMP_ID_PK	INDEX	EXAMPLE
HR	EMP_DEPARTMENT_IX	INDEX	EXAMPLE
HR	EMP_JOB_IX	INDEX	EXAMPLE
HR	EMP_MANAGER_IX	INDEX	EXAMPLE
HR	EMP_NAME_IX	INDEX	EXAMPLE

ORACLE

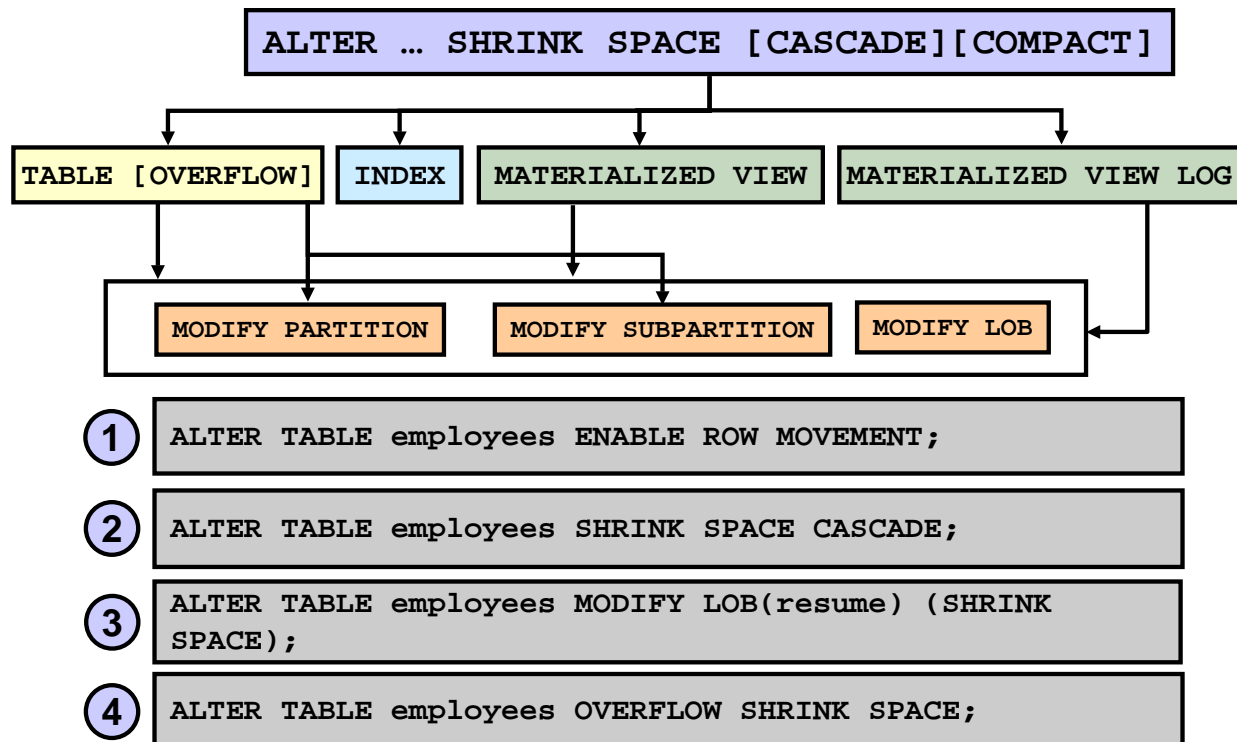
Copyright © 2009, Oracle. All rights reserved.

Manual Segment Shrink Using EM

Alternatively (to implementing the Segment Advisor recommendations), you can shrink individual segments associated with specific database objects. For example, from the Database Home page, select the Schema folder tab then click the Tables link in the Database Objects section. 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.

Shrinking Segments Using SQL



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Shrinking Segments Using SQL

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 a 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.

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.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Managing Resumable Space Allocation

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 time-out interval is associated with resumable statements. A resumable statement that is suspended for the time-out 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 only happens 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 a 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;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using Resumable Space Allocation

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 time-out 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 time-out occurs. You can also specify the time-out 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 time-out interval when using the **ENABLE RESUMABLE TIMEOUT** clause to enable resumable mode is 7,200 seconds, or 2 hours.

Using Resumable Space Allocation (continued)

You can also give a name to resumable statements. For 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.

For example:

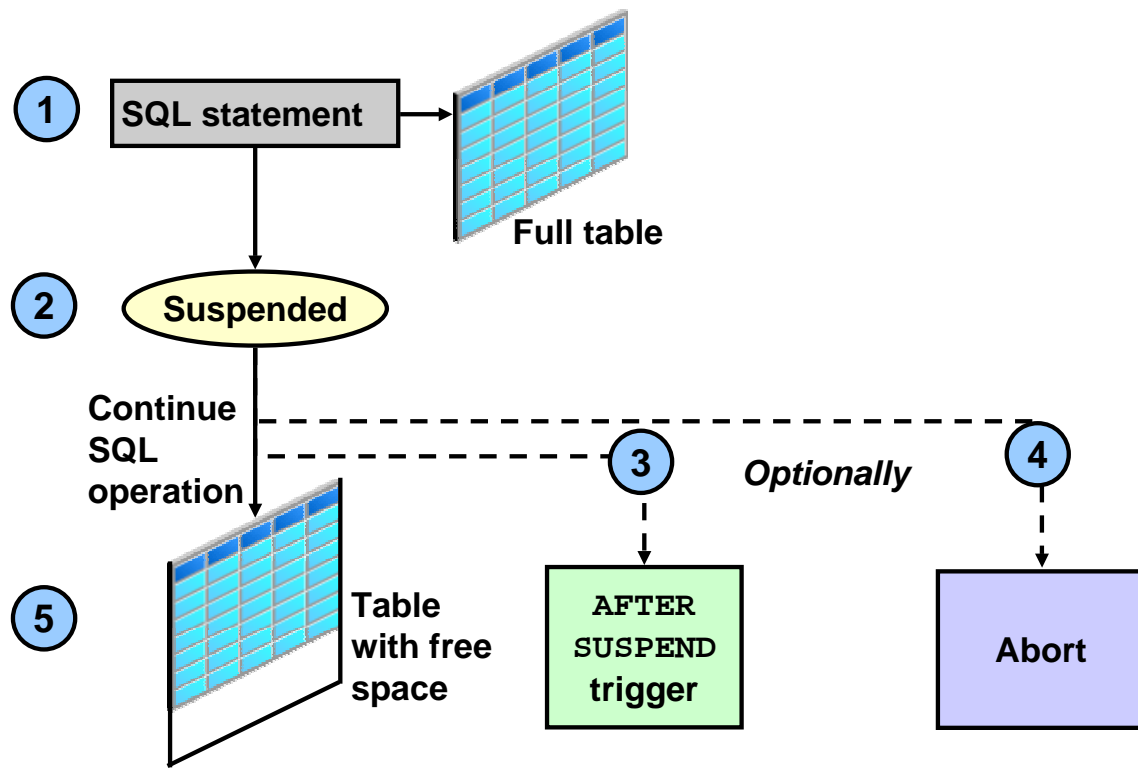
```
SELECT name, sql_text FROM user_resumable;
```

```
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 time-out 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



Copyright © 2009, Oracle. All rights reserved.

Resuming Suspending Statements

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 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.

Resuming Suspended Statements (continued)

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 time-out 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

ORACLE

Copyright © 2009, Oracle. All rights reserved.

What Operations Are Resumable?

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 `OCISStmtExecute()` and `OCISStmtFetch()` 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:

1. Segment creation in Oracle Database 11g Release2 is deferred for all tables. There are no exceptions.
2. All `UNUSABLE` indexes and index partitions are created without a segment.
3. Shrinking segment space is a nonresumable operation.
4. You can set thresholds by tablespace.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 4

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

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 18 Overview: Managing Storage

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

ORACLE

Copyright © 2009, Oracle. All rights reserved.

19

Managing Space for the Database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

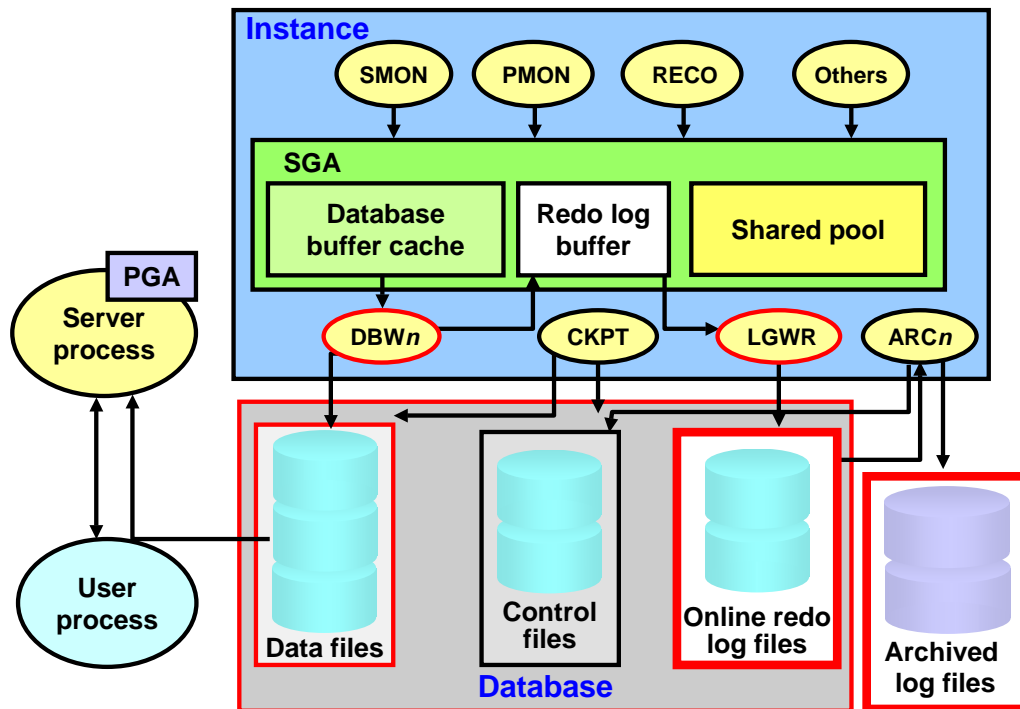
After completing this lesson, you should be able to:

- Describe the concepts and use of 4 KB-sector disks
- Use transportable tablespaces
- Describe the concepts of transportable databases

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Storage



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Storage

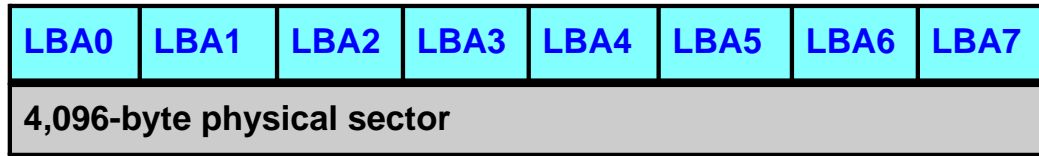
The database consists of both physical structures and logical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting access to logical storage structures.

Disks, that are a primary storage medium for database, currently have predominantly a sector size of 512 bytes, but the larger, 4 KB-sector disks are beginning to appear on the market, which offer higher storage capacity with a lower overhead. Oracle databases access the hard disk via a platform-specific device driver. (The database writer and log writer [and ASM processes] can write directly to disk without going through the OS.)

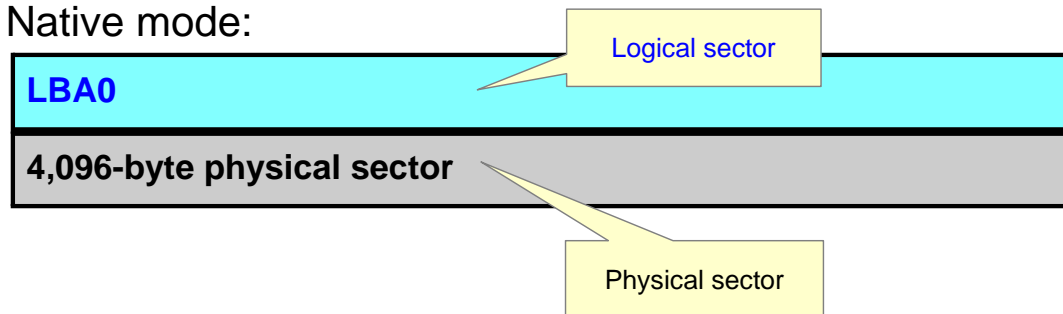
Oracle Database 11g Release2 detects the disk sector size and uses high-capacity disks without performance degradation (because of internal optimizations that reduce, for example, potential waste of redo space, which you might expect with applications such as an email system that has many short transactions).

Supporting 4-KB Sector Disks

- Emulation mode:



- Native mode:



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Supporting 4-KB Sector Disks

4-KB sector disks have physical sectors (shown in gray) and logical sectors (shown in blue). There are two types of 4-KB sector disks: emulation mode and native mode.

- 4-KB sector disks in emulation mode have eight logical sectors per one physical sector (as shown in the slide). They maintain a 512-byte interface to their 4-KB physical sectors—that is, the logical block address (LBA) references 512 bytes on disk. Performance can be decreased in emulation mode because the disk drive reads the 4 KB sector into disk cache memory, changes the 512-byte section, and then writes the entire 4 KB sector back to disk.
- 4-KB sector disks in native mode have one logical sector per physical sector (as shown in the slide). So, there is only the 4-KB interface. That is, the LBA references 4,096 bytes on disk.

Using 4-KB Sector Disks

Emulation mode:

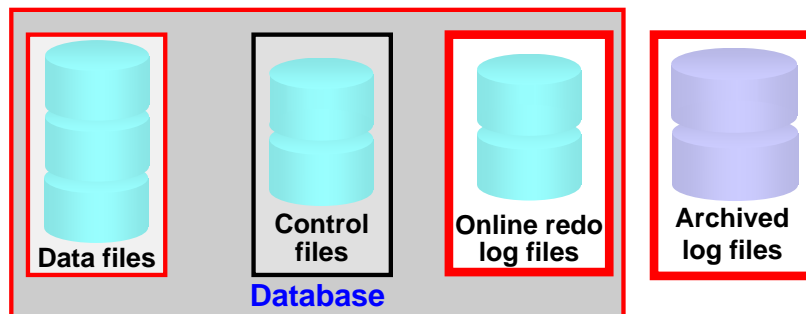
- Recommended 4-KB block size for logs
- Recommended 4-KB block size (or larger) for data files

Native mode:

- Mandatory 4-KB block size for logs
- Mandatory 4-KB block size (or larger) for data files

Not affected:

- Control file block size: 16 KB



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using 4-KB Sector Disks

In Oracle Database 11gR2, 4-KB sector disks mainly affect the redo log files. This includes online redo logs, standby redo logs, and archive logs. Oracle recommends that you create 4-KB block size logs on 4-KB emulation mode disks. On 4-KB native mode disks, you must create 4-KB block size logs.

That is, the redo block size must match the physical disk sector size (for 512-byte and for 4-KB native mode disks). Otherwise, you receive the ORA-1378 error. For 4-KB emulation mode disks, the redo block size could be 512 or 4,096 bytes. 4 KB is the preferred block size. When you are creating 512-byte blocks on a 4-KB emulation disk, a warning is printed to the alert log to indicate that the mismatched block size leads to degraded performance. This also applies to ASM disk groups.

The 4-KB sector disks also affect the Oracle data file. The Oracle database allows you the creation of 2-KB block size data files on 512-byte sector disks. With 4-KB sector disks, Oracle recommends that you create 4-KB (or larger) block size data files on the 4-KB emulation mode disks. On 4-KB native mode disks, you must create 4-KB block (or larger) size logs.

The control file block size is already 16 KB. Therefore, the 4-KB sector disks do not affect the control file.

Specifying the Disk Sector Size

Using the `SECTOR_SIZE` and `BLOCKSIZE` clauses of the following commands:

- `CREATE DISKGROUP`
- `ALTER DATABASE`
- `CREATE DATABASE`
- `CREATE CONTROL FILE`

Default sector size based on hardware (not the earlier 512-byte sectors)

```
CREATE DATABASE sample NORESETLOGS FORCE LOGGING
ARCHIVELOG
LOGFILE
  GROUP 1 '$ORACLE_BASE/oradata/sample/redo01.log'
        SIZE 100M BLOCKSIZE 4096,
  GROUP 2 '$ORACLE_BASE/oradata/sample/redo02.log'
        SIZE 100M BLOCKSIZE 4096
DATAFILE
...
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying the Disk Sector Size

In an Automatic Storage Management (ASM) environment, you can set the `SECTOR_SIZE` attribute for disk groups. This attribute can be set only at disk group creation time (by using the `CREATE DISKGROUP` command).

You can specify the size of the log file with the new `BLOCKSIZE` clause for the following commands:

- `ALTER DATABASE`
- `CREATE DATABASE`
- `CREATE CONTROL FILE`

There is no additional work for you when you create a new database on 4-KB sector disks compared to creating a new database on 512-byte disks. There is no change in the GUI environments.

You have the option of using the `BLOCKSIZE` clause in the `CREATE DATABASE` command, as shown in the slide. When you do not specify a block size, the Oracle database discovers the underlying disk sector size and uses the disk sector size as the block size for the redo log creation. So by default, the redo log block size is the disk sector size, not the earlier 512-byte sector size.

Quiz

You must use 4-KB log files on 4-KB native mode disks.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Quiz

Oracle recommends that you create 512-byte blocks on a 4-KB emulation disk for performance reasons.

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2

Quiz

Control files are not affected by 4-KB sector disks (because they are already larger).

1. True
2. False

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 1

Transporting Tablespaces

- Cross-platform transportable tablespaces:
 - Simplify moving data between data warehouse and data marts
 - Allow database migration from one platform to another
- Supported platforms include:

Solaris[tm] OE (32-bit)	HP-UX (64-bit)	Microsoft Windows IA (64-bit)
Solaris[tm] OE (64-bit)	HP Tru64 UNIX	IBM zSeries Based Linux
Microsoft Windows IA (32-bit)	HP-UX IA (64-bit)	Linux 64-bit for AMD
Linux IA (32-bit)	Linux IA (64-bit)	Apple Mac OS
AIX-Based Systems (64-bit)	HP Open VMS	Microsoft Windows 64-bit AMD
IBM Power Based Linux	HP IA Open VMS	Solaris x86 and AMD64

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Transporting Tablespaces

Transportable tablespace is the fastest way for moving large volumes of data between two Oracle databases. Using transportable tablespaces, Oracle data files (containing table data, indexes, and almost every other Oracle database object) can be directly transported from one database to another. Furthermore, like import and export, transportable tablespaces provide a mechanism for transporting metadata in addition to transporting data.

You can use the transportable tablespace feature to move data across platform boundaries. This simplifies the distribution of data from a data warehouse environment to data marts, which often run on smaller platforms. It also allows a database to be migrated from one platform to another by rebuilding the dictionary and transporting the user tablespaces.

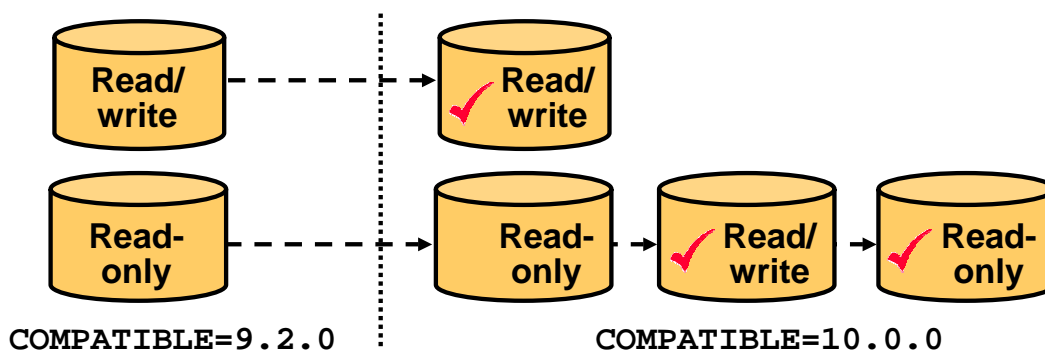
Moving data using transportable tablespaces is much faster than performing either an export/import or unload/load of the same data. This is because the data files containing all of the actual data are just copied to the destination location, and you use Data Pump to transfer only the metadata of the tablespace objects to the new database.

To be able to transport data files from one platform to another, you must ensure that both the source system and the target system are running on one of the supported platforms (see slide).

Note: The cross-platform transportable tablespace feature requires both platforms to be using the same character sets.

Concept: Minimum Compatibility Level

- Both source and target databases must have COMPATIBLE set to 10.0.0 or higher.
- Data file headers are platform-aware.
- Before transporting, make sure that all read-only and offline files are platform-aware.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Concept: Minimum Compatibility Level

Both source and target databases need to advance their database COMPATIBLE initialization parameter to 10.0.0 or greater before they can use the cross-platform transportable tablespace feature.

When data files are first opened under Oracle Database 10g or 11g with COMPATIBLE set to 10.0.0 (or greater), the files are made platform-aware. This is represented by the check marks in the diagram. Each file identifies the platform that it belongs to. These files have identical on-disk formats for file header blocks that are used for file identification and verification. Read-only and offline files get the compatibility advanced only after they are made read/write or are brought online. This implies that tablespaces that are read-only in databases before Oracle Database 10g must be made read/write at least once before they can use the cross-platform transportable feature.

Minimum Compatibility Level

	Minimum Compatibility Setting	
	Source Database	Target Database
Transport Scenario		
Databases on the same platform	8.0	8.0
Tablespace with different database block size than the target database	9.0	9.0
Databases on different platforms	10.0	10.0

ORACLE

Copyright © 2009, Oracle. All rights reserved.

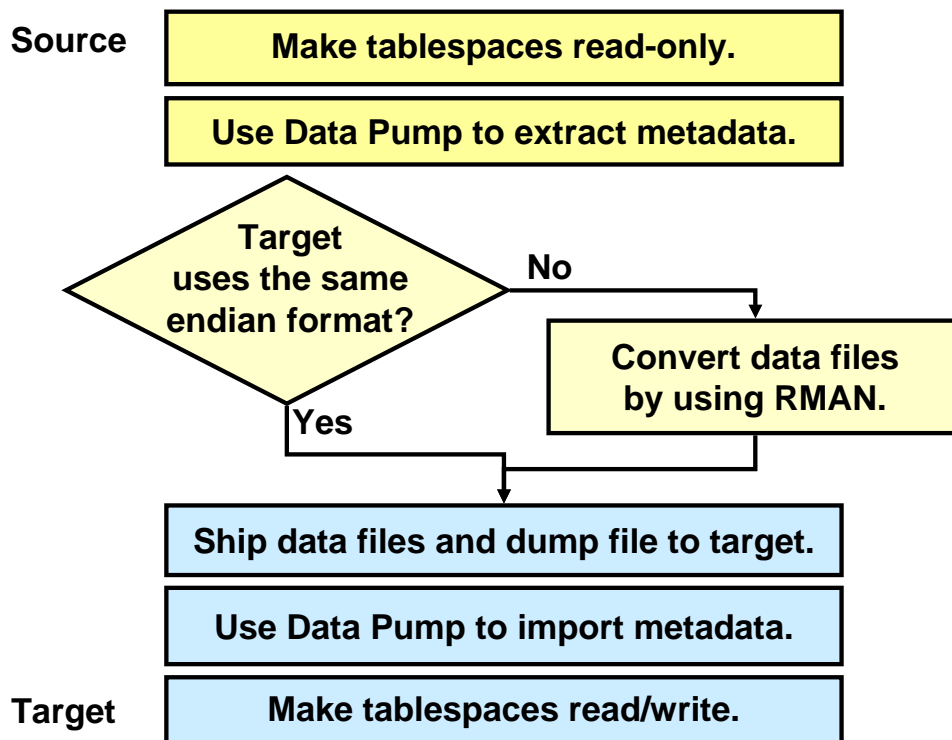
Minimum Compatibility Level

When you create a transportable tablespace set, Oracle Database computes the lowest compatibility level at which the target database must run. This is referred to as the compatibility level of the transportable set. Beginning with Oracle Database 11g, a tablespace can always be transported to a database with the same or higher compatibility setting, whether the target database is on the same or a different platform. The database signals an error if the compatibility level of the transportable set is higher than the compatibility level of the target database.

The above table shows the minimum compatibility requirements of the source and target tablespace in various scenarios. The source and target database need not have the same compatibility setting.

When data files are first opened, each file identifies the platform that it belongs to. These files have identical on-disk formats for file header blocks that are used for file identification and verification. Read-only and offline files get the compatibility advanced only after they are made read/write or are brought online.

Transportable Tablespace Procedure



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Transportable Tablespace Procedure

To transport a tablespace from one platform to another (source to target), data files belonging to the tablespace set must be converted to a format that can be understood by the target or destination database. Although with Oracle Database, disk structures conform to a common format, it is possible for the source and target platforms to use different endian formats (byte ordering). When going to a different endian platform, you must use the `CONVERT` command of the RMAN utility to convert the byte ordering. This operation can be performed on either the source or the target platforms. For platforms that have the same endian format, no conversion is needed.

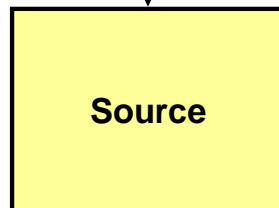
The slide graphic depicts the possible steps to transport tablespaces from a source platform to a target platform. However, it is possible to perform the conversion after shipping the files to the target platform. The last two steps must be executed on the target platform.

Basically, the procedure is the same as when using previous releases of the Oracle database server except when both platforms use different endian formats. It is assumed that both platforms are cross-transportable compliant.

Note: Byte ordering can affect the results when data is written and read. For example, the 2-byte integer value 1 is written as 0x0001 on a big-endian system (such as Sun SPARC Solaris) and as 0x0100 on a little-endian system (such as an Intel-compatible PC).

Determining the Endian Format of a Platform

```
SELECT tp.endian_format
FROM   v$transportable_platform tp, v$database d
WHERE  tp.platform_name = d.platform_name;
```



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Determining the Endian Format of a Platform

You can query `V$TRANSPORTABLE_PLATFORM` to determine whether the endian ordering is the same on both platforms. `V$DATABASE` has two columns that can be used to determine your own platform name and platform identifier. Run the query below for a comprehensive list of supported platforms and their endian formats:

```
SQL> SELECT * FROM V$TRANSPORTABLE_PLATFORM;
```

PLATFORM_ID	PLATFORM_NAME	ENDIAN_FORMAT
1	Solaris[tm] OE (32-bit)	Big
2	Solaris[tm] OE (64-bit)	Big
7	Microsoft Windows IA (32-bit)	Little
10	Linux IA (32-bit)	Little
6	AIX-Based Systems (64-bit)	Big
3	HP-UX (64-bit)	Big
5	HP Tru64 UNIX	Little
4	HP-UX IA (64-bit)	Big
11	Linux IA (64-bit)	Little
15	HP Open VMS	Little

Determining the Endian Format of a Platform (continued)

PLATFORM_ID	PLATFORM_NAME	ENDIAN_FORMAT
-----	-----	-----
8	Microsoft Windows IA (64-bit)	Little
9	IBM zSeries Based Linux	Big
13	Linux 64-bit for AMD	Little
16	Apple Mac OS	Big
12	Microsoft Windows 64-bit for AMD	Little
17	Solaris Operating System (x86)	Little
18	IBM Power Based Linux	Big
19	HP IA Open VMS	Little
20	Solaris Operating System (AMD64)	Little

Using the RMAN CONVERT Command

RMAN:

- Converts tablespaces, data files, or databases to the format of a destination platform
- Does not change input files
- Writes converted files to output destination

```
CONNECT TARGET SYS@orcl
RMAN>
SQL 'ALTER TABLESPACE hr READ ONLY';
CONVERT TABLESPACE hr
      TO PLATFORM 'Solaris[tm] OE (64-bit)'
      FORMAT '/tmp/transport_to_solaris/%U';;
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the RMAN CONVERT Command

You use the RMAN CONVERT command to convert a tablespace, data file, or database to the format of a destination platform in preparation for transport across different platforms. Input files are not altered by CONVERT because the conversion is not performed in place. Instead, RMAN writes converted files to a specified output destination.

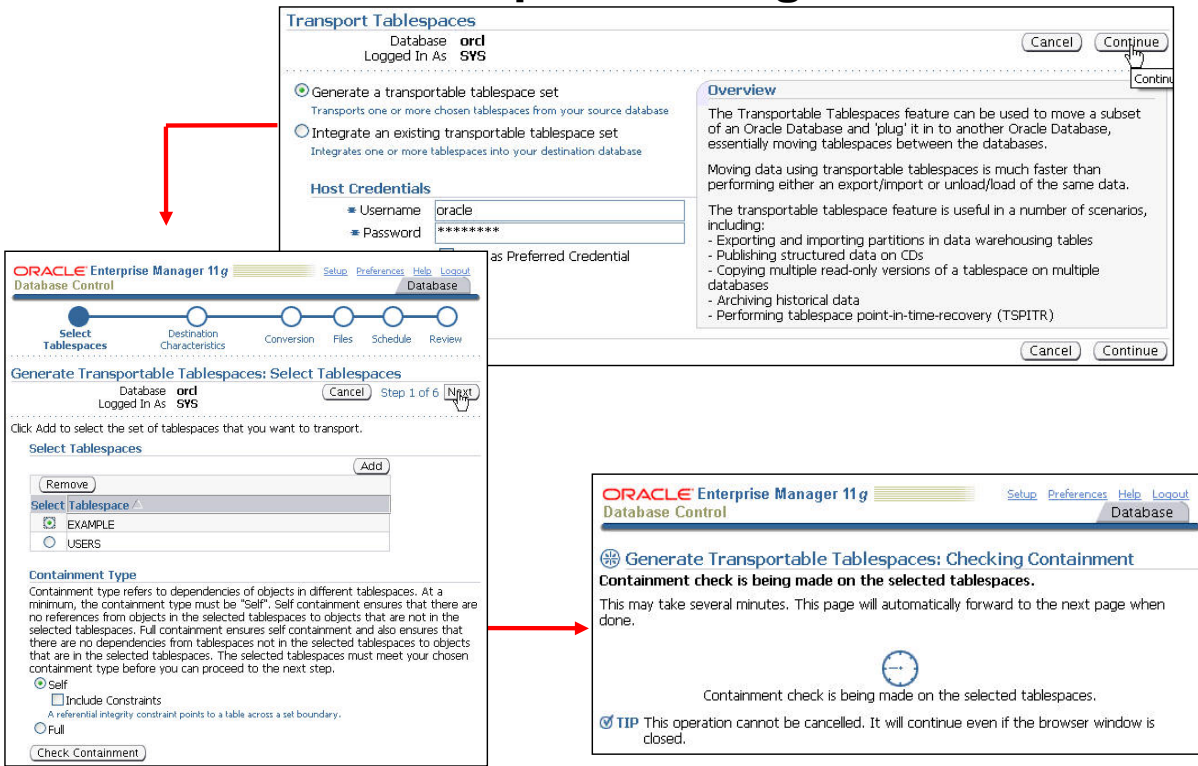
CONVERT TABLESPACE example:

- Assume that you have an ORCL database on a Linux 32-bit platform, which you want to transport to a Solaris 64-bit platform.
- Connect as TARGET to the *source* database (mounted or open).
- The tablespace must be read-only at the time of conversion.
- The result is a set of converted data files in the `/tmp/transport_to_solaris/` directory, with data in the right endian-order for the Solaris 64-bit platform.

Restrictions: The CONVERT command does not process user data types that require endian conversions. To transport objects between databases that are built on underlying types that store data in a platform-specific format, use the Data Pump Import and Export utilities.

For detailed prerequisites, usage, restrictions, and syntax, see the *Oracle Database Backup and Recovery Reference*.

Transportable Tablespaces with Enterprise Manager



Copyright © 2009, Oracle. All rights reserved.

Transportable Tablespaces with Enterprise Manager

Enterprise Manager can be used to implement transportable tablespaces. From the Database home page, click the Data Movement folder tab, and then click Transport Tablespaces under the Move Database Files section. Select “Generate a transportable table set” and provide the login credentials for the `oracle` user, and then click Continue. On the Select Tablespaces page, add the tablespaces you want to transport from the displayed list by clicking the Tablespace button, and then clicking Add. Near the bottom of the page, you must select the level of containment checking to be done before the tablespaces are processed. The containment check looks for object dependencies within the tablespaces. When you have finished, click Next. Wait a few moments while the containment check runs. Address any issues found by the check before continuing.

Transportable Tablespaces with Enterprise Manager

ORACLE Enterprise Manager 11g Database Control

Generate Transportable Tablespaces: Destination Characteristics

Database: orcl
Logged In As: SYS
Source Platform: Linux IA (32-bit)

Cancel Back Step 2 of 6 Next

Destination Database Platform

If the source platform is different from the destination platform, there may be a need to convert the source datafiles to the destination format. Specify the destination platform to check for datafile format compatibility.

Destination Platform: Linux IA (32-bit)

The Conversion step will be skipped if the destination platform is compatible.

Destination Character Set

For the tablespaces to be successfully integrated at the destination database, the source destination database must have compatible character sets. Specify the character set of the destination database to check for compatibility.

☒ Verify Compatibility of Destination Character Sets

Destination Database Character Set: AL32UTF8

Destination National Character Set: AL16UTF16

ORACLE Enterprise Manager 11g Database Control

Generate Transportable Tablespaces: Files

Database: orcl
Logged In As: SYS

Cancel Finish Back Step 4 of 6 Next

Dump File

Specify the location where the dump file associated with the transportable tablespace can be generated.

* Dump File Directory: /u01/app/oracle/oradata/orcl/

* Dump File Name: EXPDAT_GENERATETTS000041.DMP

Datafiles

A copy of the datafiles associated with the selected tablespaces is needed. Selected tablespaces will be made read-only as part of Generate Transportable Tablespaces operation.

☒ Copy Datafiles Automatically (Recommended)

The datafiles will be copied to the above specified dump file directory. This operation will require 105 MB of free disk space. The selected tablespaces will be made read-only for the duration of the job.

☐ Manually Copy Datafiles

You are responsible for copying the datafiles. Tablespaces remain in read-only mode after the job is completed to maintain datafile consistency with the dump file. You must manually put tablespaces back into read-write mode after copying datafiles.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Transportable Tablespaces with Enterprise Manager (continued)

On the Destination Characteristics page, you must supply the destination platform and character sets. Under the Destination Database Platform section, select the operating system of the destination machine from the drop-down list. If the destination platform is different from the source platform, Enterprise Manager will perform a data conversion. Continue to the Destination Character Set section of the page and choose the destination character set and national character set from the drop-down lists. These character sets must be compatible with the source sets. When you click Next to continue, Enterprise Manager checks the compatibility of the character sets. If the chosen character sets are flagged as incompatible, you will be returned back to the Destination Characteristics page to correct your selections.

Transportable Tablespaces with Enterprise Manager

ORACLE Enterprise Manager 11g
Database Control

Setup Preferences Help Logout Database

Select Tablespaces Destination Characteristics Conversion Files **Schedule** Review

Generate Transportable Tablespaces: Schedule

Database: orcl
Logged In As: SYS

Cancel Finish Back Step 5 of 6 Next

Specify a name and description for the export job. Specify a date to start the job.

Job Parameters

Job Name: GENERATETTS000041
Description: Transport Tablespace(s)

Job Schedule

Start: ☒ Immediately

Generate Transportable Tablespaces: Review

Database: orcl
Logged In As: SYS

Cancel Back Step 6 of 6 Submit Job

Summary

Job Name: GENERATETTS000041
Transport Tablespaces: EXAMPLE,USERS
Containment Type: SELF
Include Constraints: false
Source Platform: Linux IA (32-bit)
Destination Platform: Linux IA (32-bit)
Disk Space Required (MB): 105
Copy the datafiles: AUTO
Dump File Format: Datapump

Dump File

Dump File Directory: /u01/app/oracle/oradata/orcl/
Dump File Name: EXPDAT_GENERATETTS000041.DMP

Datafiles

Datafile Name	New Datafile Name	Size (MB)
/u01/app/oracle/oradata/orcl/users01.dbf	/u01/app/oracle/oradata/orcl/users01.dbf	5
/u01/app/oracle/oradata/orcl/example01.dbf	/u01/app/oracle/oradata/orcl/example01.dbf	100

Cancel Back Step 6 of 6 Submit Job

Database | Setup | Preferences | Help | Logout

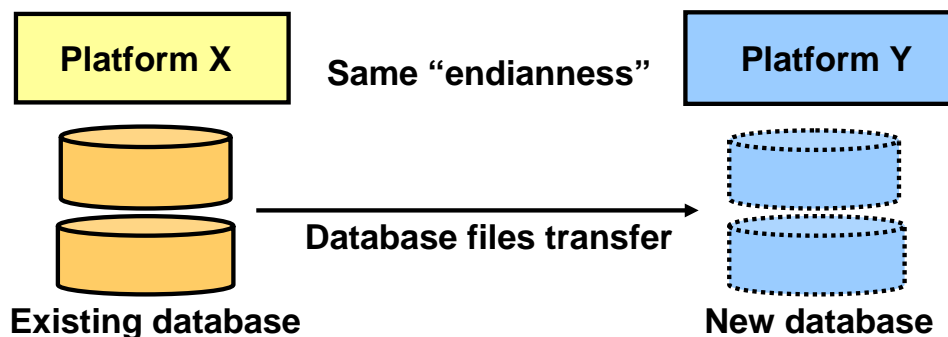
Copyright © 2009, Oracle. All rights reserved.

Transportable Tablespaces with Enterprise Manager (continued)

On the Schedule page, supply a meaningful description for the default job name. You can also choose to start the job immediately or schedule it for later execution. When you have made your selections, click Next to continue. On the review page, you can verify your choices before submitting the job for execution. Click the Submit Job button if the entries are correct. Click the Back button to correct any incorrect entries.

Transporting Databases

- Generalize the transportable tablespace feature.
- Data subsets can easily be distributed from a data warehousing environment to data marts, which are usually on smaller platforms.
- A database can be migrated from one platform to another very quickly.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Transporting Databases

You can use the transportable tablespace feature to migrate a database to a different platform by creating a new database on the destination platform and performing a transport of all the user tablespaces. You cannot transport the `SYSTEM` tablespace. Therefore, objects such as sequences, PL/SQL packages, and other objects that depend on the `SYSTEM` tablespace are not transported. You must either create these objects manually on the destination database, or use Data Pump to transport the objects that are not moved by transportable tablespace.

To transport databases from one platform to another, you must ensure that both the source system and the target system are running on one of the platforms that are listed in `V$TRANSPORTABLE_PLATFORM` and that both have the same endian format. For example, you can transport a database running on Linux IA (32-bit) to one of the Windows platforms.

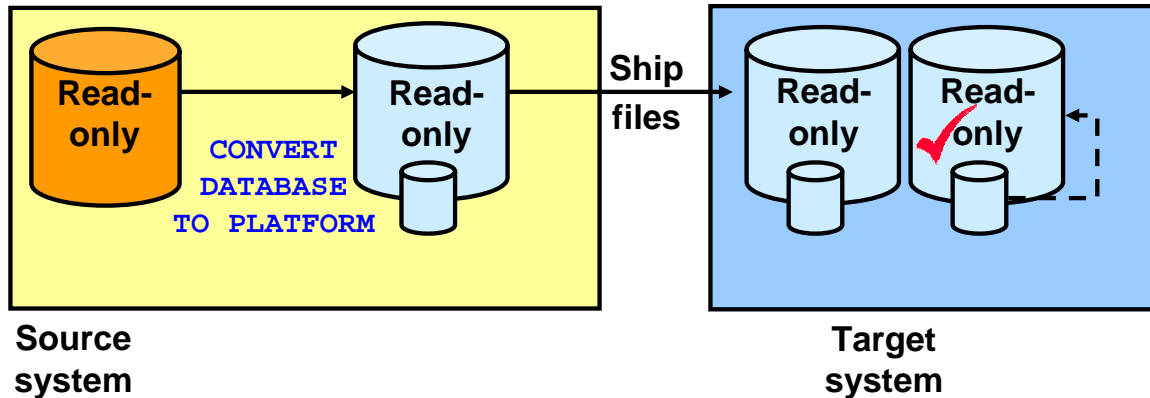
If one or both of the databases uses Automatic Storage Management (ASM), you may need to use the `DBMS_FILE_TRANSFER` package to FTP the files.

Unlike transportable tablespace, where there is a target database to plug data into, this feature creates a new database on the target platform. The newly created database contains the same data as the source database. Except for things such as database name, instance name, and location of files, the new database also has the same settings as the source database.

Note: Transporting database is faster than using Data Pump to move data.

Database Transportation Procedure: Source System Conversion

Open database in **READ ONLY** mode
and **COMPATIBLE=10.0.0** or higher.



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Transportation Procedure: Source System Conversion

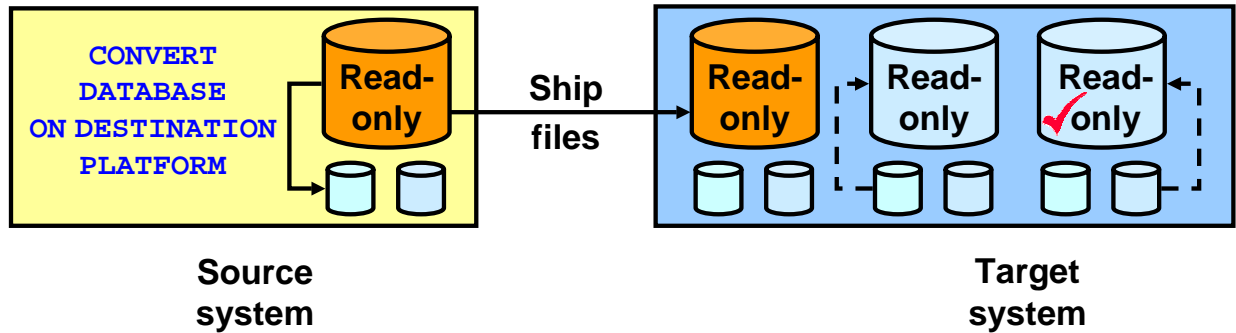
Before you can transport your database, you must open it in **READ ONLY** mode. Then use RMAN to convert the necessary data files of the database.

When you do the conversion on the source platform, the **RMAN CONVERT DATABASE** command generates a script containing the correct **CREATE CONTROLFILE RESETLOGS** command that is used on the target system to create the new database. The **CONVERT DATABASE** command then converts all identified data files so that they can be used on the target system. You then ship the converted data files and the generated script to the target platform. By executing the generated script on the target platform, you create a new copy of your database.

Note: The source database must be running with the **COMPATIBLE** initialization parameter set to **10.0.0** or higher. All identified tablespaces must have been **READ WRITE** at least once since the time that **COMPATIBLE** was set to **10.0.0** or higher.

Database Transportation Procedure: Target System Conversion

Open database in **READ ONLY** mode
and **COMPATIBLE=10.0.0** or higher



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Transportation Procedure: Target System Conversion

Before you can transport your database, you must open it in **READ ONLY** mode. Then use RMAN to convert the necessary data files of the database.

When you do the conversion on the target platform, the **CONVERT DATABASE** command (which is executed on the source system) generates only two scripts used on the target system to convert the data files, and to re-create the control files for the new database. Then, you ship the identified data files and both scripts to the target platform. After this is done, execute both scripts in the right order. The first one uses the existing **RMAN CONVERT DATAFILE** command to do the conversion, and the second issues the **CREATE CONTROLFILE RESETLOGS SQL** command with the converted data files to create the new database.

Note: The source database must be running with the **COMPATIBLE** initialization parameter set to **10.0.0** or higher. All identified tablespaces must have been **READ WRITE** at least once since **COMPATIBLE** was set to **10.0.0** or higher.

Database Transportation: Considerations

- Create the password file on the target platform.
- Transport the BFILES used in the source database.
- The generated PFILE and transport script use OMF.
- Use DBNEWID to change the DBID.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Database Transportation: Considerations

Redo logs, control files, and tempfiles are not transported. They are re-created for the new database on the target platform. As a result, the new database on the target platform must be opened with the RESETLOGS option.

If a password file is used, it is not transported and you need to create it on the target platform. This is because the types of file names allowed for the password file are OS specific. However, the output of the CONVERT DATABASE command lists all the usernames and their system privileges, and advises to re-create the password file and add entries for these users on the target platform.

The CONVERT DATABASE command lists all the directory objects and objects that use BFILE data types or external tables in the source database. You may need to update these objects with new directory and file names. If BFILES are used in the database, you have to transport the BFILES.

The generated PFILE and transport script use Oracle Managed Files (OMF) for database files. If you do not want to use OMF, you must modify the PFILE and transport script.

The transported database has the same DBID as the source database. You can use the DBNEWID utility to change the DBID. In the transport script as well as the output of the CONVERT DATABASE command, you are prompted to use the DBNEWID utility to change the database ID.

Quiz

Select the statements that are true:

1. The RMAN `CONVERT` command performs an in-place conversion, so your input files are changed before they are transported to the destination.
2. Read/write tablespaces need to be in read-only mode at the time of an endian conversion.
3. You can use the RMAN `CONVERT` command for tables, tablespaces, and databases.
4. You can transport databases into a data warehouse environment.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 4

Summary

In this lesson, you should have learned how to:

- Describe the concepts and use of 4 KB-sector disks
- Describe the concepts of transportable tablespaces and databases

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 19 Overview: Managing Space for the Database

This practice covers the following topic:

- Viewing a demonstration on “Using 4 KB-sector disks”

ORACLE

Copyright © 2009, Oracle. All rights reserved.

20

Duplicating a Database

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- List the purposes of creating a duplicate database
- Choose a technique for duplicating a database
- Duplicate a database with RMAN
- Use an RMAN backup to duplicate a database
- Duplicate a database based on a running instance

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using a Duplicate Database

- Using a duplicate database to:
 - Test backup and recovery procedures
 - Recover objects by creating an export and importing the objects into the production database
- Creating a duplicate database:
 - With the RMAN `DUPLICATE` command
 - On the same or separate hosts
 - With the identical content, or subset of source
 - Performed by auxiliary channels for backup-up based duplication
 - Performed by target channels for active database duplication

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using a Duplicate Database

A duplicate database is a copy of your target database with a new, unique database identifier (DBID). You can operate it independently of the target database to:

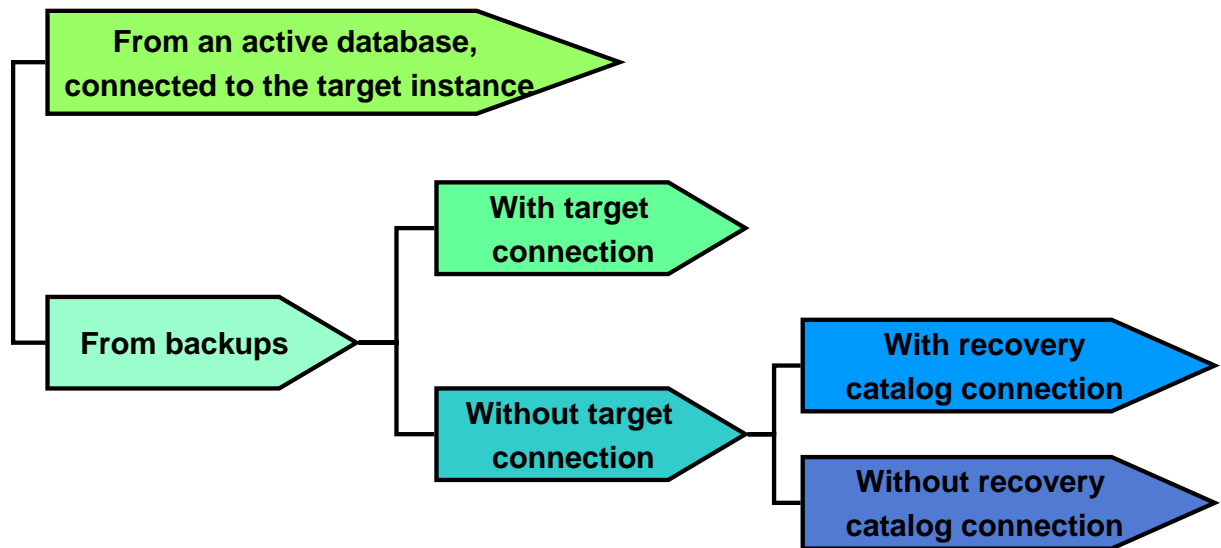
- Test backup and recovery procedures
- Recover objects that were inadvertently dropped from the target database by creating an export containing the objects in the duplicate database and importing them into the production database. Although you probably find that Flashback Query, Flashback Drop, and Flashback Table are a much easier and faster solution to recover objects.

Creating a duplicate database:

- You can use the RMAN `DUPLICATE` command to create a duplicate database on the same host or separate hosts.
- The duplicate database can include the same content or only a subset from the source database (more details later in this lesson).
- The principal work of the duplication is performed by the auxiliary channels. These channels correspond to a server session on the auxiliary instance on the destination host for backup-based duplication.
- For active database duplication, the target channels perform the work of pushing data file copies to the auxiliary instance.

Choosing Database Duplication Techniques

Choosing a technique to duplicate your database—always with connection to the auxiliary instance:



ORACLE

Copyright © 2009, Oracle. All rights reserved.

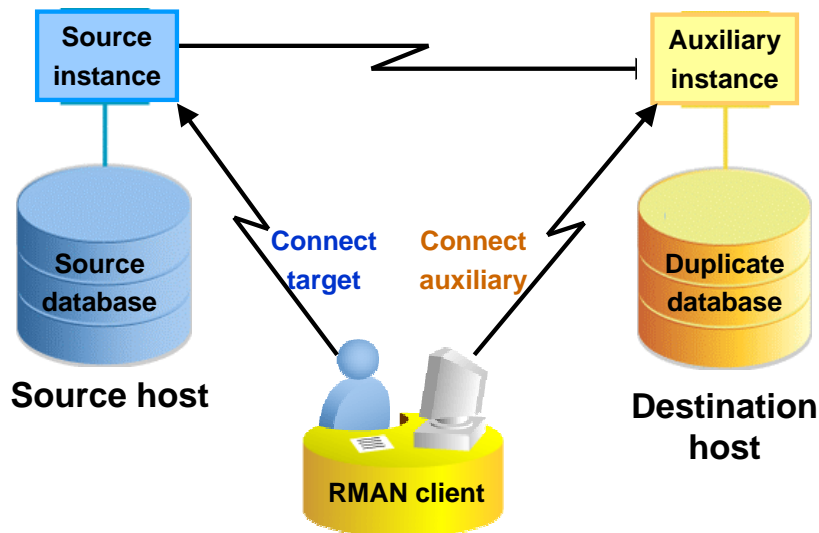
Database Duplication Techniques

You can duplicate a source database to a destination database, which can be on the same or different computers. The database instance associated with the duplicate database is called the auxiliary instance. All duplication techniques require a connection to the auxiliary instance. The diagram shows you the following techniques for database duplication:

- From an active database, connected to the target and auxiliary instances
- From backup, connected to the target and auxiliary instances
- From backup, connected to the auxiliary instance, not connected to the target, but with recovery catalog connection
- From backup, connected to the auxiliary instance, not connected to the target and the recovery catalog

Duplicating an Active Database

- With network (no backups required)
- Including customized `SPFILE`
- Via Enterprise Manager or RMAN command line



ORACLE

Copyright © 2009, Oracle. All rights reserved.

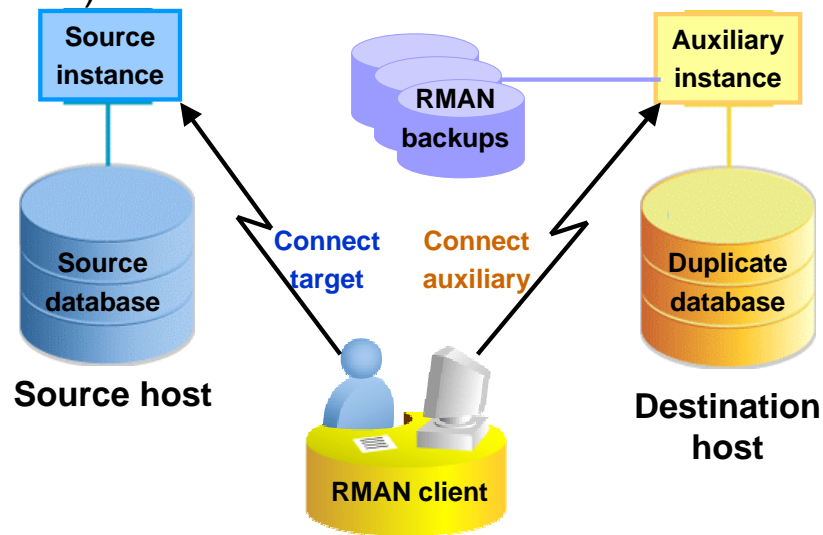
Duplicating an Active Database

You can instruct the source database to perform online image copies and archived log copies directly to the auxiliary instance by using Enterprise Manager or the `FROM ACTIVE DATABASE` clause of the `RMAN DUPLICATE` command. Backups are not needed for this operation. RMAN connects as `TARGET` to the source database instance and as `AUXILIARY` to the auxiliary instance (as shown in the slide).

The database files are copied from the source to a destination or auxiliary instance via an interinstance network connection. RMAN then uses a “memory script” (one that is contained only in memory) to complete recovery and open the database.

Duplicating a Database with a Target Connection

- Connecting to the target (source database)
- Connecting to the auxiliary instance
- Optionally, connecting to the recovery catalog (or using target control file)



Copyright © 2009, Oracle. All rights reserved.

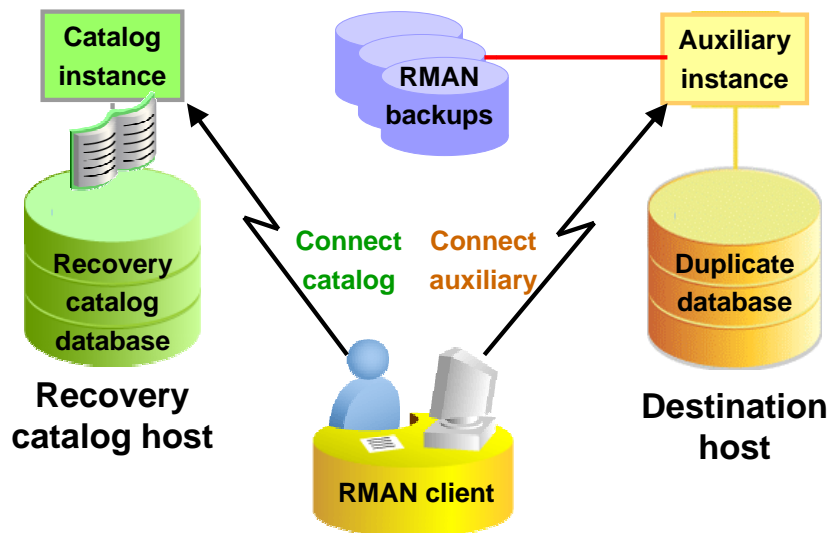
Duplicating a Database with a Target Connection

When you duplicate a database with a target database connection, RMAN can obtain metadata about backups either from the target database control file or from the recovery catalog.

The diagram illustrates backup-based duplication with a target connection. RMAN connects to the source database instance and the auxiliary instance. Optionally, RMAN can connect to a recovery catalog database (not shown in the graphic). The destination host must have access to the RMAN backups required to create the duplicate database.

Duplicating a Database with Recovery Catalog Without Target Connection

- Connecting to a recovery catalog for backup metadata
- Connecting to the auxiliary instance, which must have access to the RMAN backups



ORACLE

Copyright © 2009, Oracle. All rights reserved.

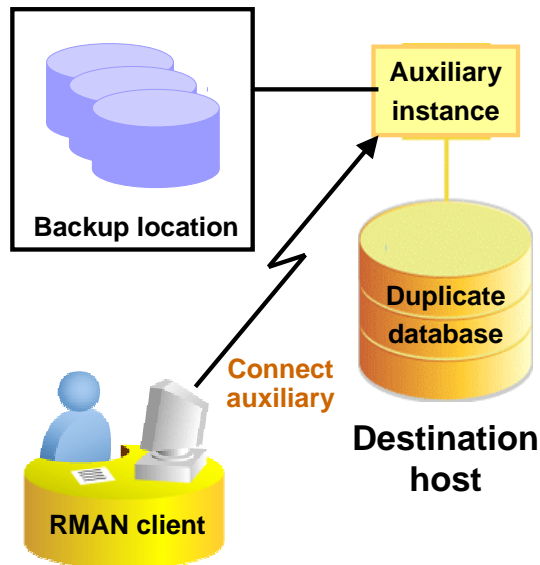
Duplicating a Database Without Target Connection

When you duplicate a database without a target database connection, but with a recovery catalog, RMAN uses the recovery catalog to obtain metadata about the backups.

The diagram illustrates backup-based duplication without a target connection. RMAN connects to a recovery catalog database instance and the auxiliary instance. The destination host must have access to the RMAN backups required to create the duplicate database.

Duplicating a Database Without Recovery Catalog or Target Connection

Connecting to the auxiliary instance, which must have access to a disk `BACKUP LOCATION`



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Duplicating a Database Without Recovery Catalog or Target Connection

When you duplicate a database without a target database connection and without a recovery catalog, RMAN uses a `BACKUP LOCATION` where all necessary backups and copies reside.

The diagram illustrates backup-based duplication without connections to the target or to the recovery catalog database instance. A disk backup location containing all the backups or copies for duplication must be available to the destination host.

Creating a Backup-Based Duplicate Database

1. Create an Oracle password file for the auxiliary instance.
2. Establish Oracle Net connectivity to the auxiliary instance.
3. Create an initialization parameter file for the auxiliary instance.
4. Start the auxiliary instance in `NOMOUNT` mode.
5. Mount or open the target database.
6. Ensure that backups and archived redo log files are available.
7. Allocate auxiliary channels if needed.
8. Execute the `DUPLICATE` command.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating a Backup-Based Duplicate Database

It is important to understand these basic steps and the RMAN database duplication process.

If you are using the Enterprise Manager interface, wizards can perform most steps for you. If you are creating a duplicate database with the command-line interface, you need to perform the steps manually. You can also use the EM interface as a test or sample, and use the output log as a basis for scripting your own database duplication.

The basic steps for creating a duplicate database are outlined in the slide. More detail is provided in this lesson for some of the steps.

Creating an Initialization Parameter File for the Auxiliary Instance

Specify parameters as follows:

- **DB_NAME**
 - If the duplicate database is in the same Oracle home as the target database, names must be different.
 - Use the same value in the `DUPLICATE` command.
- **DB_BLOCK_SIZE**
 - Specify the same value as set for the target database.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Creating an Initialization Parameter File for the Auxiliary Instance

You must create a text initialization parameter file for the auxiliary instance. The text initialization parameter file must reside on the same host as the RMAN client that you use to execute the `DUPLICATE` command.

Take note of the requirements for each of the following parameters:

- **DB_NAME:** If the target database and the duplicate database are in the same Oracle home, you must set `DB_NAME` to a different name. If they are in different Oracle homes, you must ensure that the name of the duplicate database differs from the other names in its Oracle home. Be sure to use the same database name that you set for this parameter when you execute the `DUPLICATE` command.
- **DB_BLOCK_SIZE:** The block size of the auxiliary database must match the block size of the target database. Specify the same value in the initialization parameter file for the auxiliary database as set in the initialization parameter file for the target database. If the parameter is not set in the initialization parameter file for the target database, do not set it in the auxiliary instance initialization parameter file.

In addition, be sure to verify the settings of all initialization parameters that specify path names. Verify that all specified paths are accessible on the duplicate database host.

Specifying New Names for Your Destination

Available techniques:

- SET NEWNAME command
- CONFIGURE AUXNAME command (deprecated for recovery set data files)
- DB_FILE_NAME_CONVERT parameter with the DUPLICATE command

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying New Names for Your Destination

You can use the following techniques to specify new names for data files:

- Include the SET NEWNAME FOR DATAFILE command within a RUN block to specify new names for the data files.
- Use the CONFIGURE AUXNAME command.

CONFIGURE AUXNAME is an alternative to SET NEWNAME. The difference is that after you configure the auxiliary name the first time, additional DUPLICATE commands reuse the configured settings. In contrast, you must reissue the SET NEWNAME command every time you execute the DUPLICATE command.

Note: SET NEWNAME replaces CONFIGURE AUXNAME for recovery set data files

- Specify the DB_FILE_NAME_CONVERT parameter with the DUPLICATE command.

Using the SET NEWNAME Clauses

- SET NEWNAME clauses enable you to specify a default name format for all files in a database or in a named tablespace.
- The default name is used for DUPLICATE, RESTORE, and SWITCH commands in the RUN block.
- It enables you to set file names with a single command rather than setting each file name individually.

```
SET NEWNAME FOR DATABASE  
TO {NEW| 'formatSpec'};
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using the SET NEWNAME Clauses

You can use SET NEWNAME to specify the default name format for all data files in a named tablespace and all data files in the database.

The order of precedence for the SET NEWNAME command is as follows:

1. SET NEWNAME FOR DATAFILE and SET NEWNAME FOR TEMPFILE
2. SET NEWNAME FOR TABLESPACE
3. SET NEWNAME FOR DATABASE

Example:

```
RUN  
{  
SET NEWNAME FOR DATABASE TO '/u01/app/oracle/oradata/duplodb/%b';  
DUPLICATE TARGET DATABASE TO duplodb  
LOGFILE  
GROUP 1 ('/u01/app/oracle/oradata/duplodb/redo01a.log',  
'/u01/app/oracle/oradata/duplodb/redo01b.log') SIZE 50M REUSE,  
GROUP 2 ('/u01/app/oracle/oradata/duplodb/redo02a.log',  
'/u01/app/oracle/oradata/duplodb/redo02b.log') SIZE 50M REUSE,  
GROUP 3 ('/u01/app/oracle/oradata/duplodb/redo03a.log',  
'/u01/app/oracle/oradata/duplodb/redo03b.log') SIZE 50M REUSE;  
}
```

Substitution Variables for SET NEWNAME

Syntax Element	Description
%b	Specifies the file name without the directory path
%f	Specifies the absolute file number of the data file for which the new name is generated
%I	Specifies the DBID
%N	Specifies the tablespace name
%U	Specifies a system-generated file name of the format: data-D-%d_id-%I_TS-%N_FNO-%f

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Substitution Variables for SET NEWNAME

When issuing SET NEWNAME FOR DATABASE or SET NEWNAME FOR TABLESPACE, you must specify substitution variables in the TO <filename> clause to avoid name collisions. Specify at least one of the following substitution variables: %b, %f, and %U. %I and %N are optional variables.

Specifying Parameters for File Naming

Alternatively, specify the following parameters to explicitly control the naming of the files of your auxiliary database:

- CONTROL_FILES
- DB_FILE_NAME_CONVERT
- LOG_FILE_NAME_CONVERT

```
CONTROL_FILES='/u01/app/oracle/oradata/aux/control01.ctl',
              '/u01/app/oracle/oradata/aux/control02.ctl',
              '/u01/app/oracle/oradata/aux/control03.ctl'
DB_FILE_NAME_CONVERT='/u01/app/oracle/oradata/orcl',
                    '/u01/app/oracle/oradata/aux'
LOG_FILE_NAME_CONVERT='/u01/app/oracle/oradata/orcl',
                     '/u01/app/oracle/oradata/aux'
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying Parameters for File Naming

RMAN generates names for the required database files when you execute the DUPLICATE command. You can control the naming of the files by specifying the following initialization parameters in the auxiliary instance initialization parameter file:

- **CONTROL_FILES:** Specify the names of the control files in this parameter. If you do not set the names via this parameter, the Oracle server creates an Oracle-managed control file in a default control destination. Refer to the SQL CREATE CONTROLFILE command in the SQL Reference manual for specific information.
- **DB_FILE_NAME_CONVERT:** This parameter is used to specify the names of data files for the auxiliary database. It has the format DB_FILE_NAME_CONVERT = '*string1*' , '*string2*', where *string1* is the pattern of the target database file name and *string2* is the pattern of the auxiliary database file name. You can also specify the DB_FILE_NAME_CONVERT parameter as an option to the DUPLICATE DATABASE command.
- **LOG_FILE_NAME_CONVERT:** This parameter is used to specify the names of the redo log files for the auxiliary database. It has the format LOG_FILE_NAME_CONVERT = '*string1*' , '*string2*', where *string1* is the pattern of the target database file name and *string2* is the pattern of the auxiliary database file name. You can also use the LOGFILE clause of the DUPLICATE DATABASE command to specify redo log file names.

Specifying Parameters for File Naming (continued)

As an alternative to using the initialization parameters to control the naming of the files, you can use the following techniques to rename the redo log files:

- Use the LOGFILE clause of the DUPLICATE command.
- Set the Oracle Managed Files initialization parameters: DB_CREATE_FILE_DEST, DB_CREATE_ONLINE_DEST_ *n*, or DB_RECOVERY_FILE_DEST.

Starting the Instance in NOMOUNT Mode

- Start the auxiliary instance in NOMOUNT mode.
- Create a server parameter file (SPFILE) from the text initialization parameter file you used to start the instance.

```
SQL> startup nomount pfile='$HOME/auxinstance/initAUX.ora'
ORACLE instance started.

Total System Global Area  285212672 bytes
Fixed Size                  1218992 bytes
Variable Size              92276304 bytes
Database Buffers           188743680 bytes
Redo Buffers                2973696 bytes
SQL> create spfile
      2  from pfile='$HOME/auxinstance/initAUX.ora';

File created.
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Starting the Instance in NOMOUNT Mode

After you have created the text initialization parameter file, invoke SQL*Plus to start the auxiliary instance in NOMOUNT mode.

After you invoke SQL*Plus, create a server parameter file (SPFILE) from your text initialization parameter file. You can execute the CREATE SPFILE before or after you have started the instance. You should create the SPFILE in the default location so that you do not need to specify the PFILE option with the DUPLICATE command. RMAN shuts down the auxiliary instance and restarts it as part of the duplication process, so you must specify the PFILE option if you do not use an SPFILE.

Ensuring That Backups and Archived Redo Log Files Are Available

- Backups of all target database data files must be accessible on the duplicate host.
- Backups can be a combination of full and incremental backups.
- Archived redo log files needed to recover the duplicate database must be accessible on the duplicate host.
- Archived redo log files can be:
 - Backups on a media manager
 - Image copies
 - Actual archived redo log files

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Ensuring That Backups and Archived Redo Log Files Are Available

The backups needed to restore the data files must be accessible on the duplicate host. You do not need a whole database backup. RMAN can use a combination of full and incremental backups of individual data files during the duplication process.

Archived redo logs required to recover the duplicate database to the desired point in time must also be accessible. The archived redo log files can be backups, image copies, or the actual archived redo logs. The backups or copies can be transferred to the local disk of the duplicate database node or mounted across a network by some means such as network file system (NFS).

Allocating Auxiliary Channels

- Auxiliary channels specify a connection between RMAN and an auxiliary database instance.
- If automatic channels are not configured, allocate auxiliary channels:
 - Start RMAN with a connection to the target database instance, the auxiliary instance, and recovery catalog if applicable.
 - Allocate at least one auxiliary channel within the RUN block.

```
$ rman target sys/oracle_4U@trgt auxiliary  
sys/oracle_4U@auxdb  
RMAN> RUN  
    {ALLOCATE AUXILIARY CHANNEL aux1 DEVICE TYPE DISK;  
    ALLOCATE AUXILIARY CHANNEL aux2 DEVICE TYPE DISK;  
    ...  
    DUPLICATE TARGET DATABASE to auxdb; . . .
```

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Allocating Auxiliary Channels

If you do not have automatic channels configured, manually allocate at least one auxiliary channel before issuing the DUPLICATE command. The ALLOCATE AUXILIARY CHANNEL command must be within the same RUN block as the DUPLICATE command.

The channel type specified on the ALLOCATE AUXILIARY CHANNEL command must match the media where the backups of the target database are located.

- If the backups reside on disk, you can allocate more than one channel to reduce the time it takes for the duplication process.
- For tape backups, you can specify the number of channels that correspond to the number of devices available.

The auxiliary instance must be started with the NOMOUNT option and the target database must be mounted or open.

Understanding the RMAN Duplication Operation

When you execute the `DUPLICATE` command, RMAN performs the following operations:

- 1A. Creates a control file server parameter file for the auxiliary instance (for active and for backup-based duplication with target connection) , *or*:
- 1B. Restores from backup (for standby database and for backup-based duplication without target connection)
2. Mounts the backup control file
3. For backup-based duplication: Selects the backups for restoring the data files to the auxiliary instance
4. Restores the target data files to the duplicate database
5. Performs incomplete recovery using all available incremental backups and archived redo log files

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Understanding the RMAN Duplication Operation

When you execute the `DUPLICATE` command, RMAN performs the operations listed in the slide.

- 1A. RMAN creates a default server parameter file for the auxiliary instance if the following conditions are true:
 - Duplication does not involve a standby database.
 - Server parameter files are not being duplicated.
 - The auxiliary instance was not started with a server parameter file.
- 1B. RMAN restores from backup—always for the standby database, and for backup-based duplication without target connection.
2. RMAN mounts the restored or the copied backup control file from the active database.
3. For backup-based duplication: RMAN uses the RMAN repository to select the backups for restoring the data files to the auxiliary instance.
4. RMAN restores and copies the duplicate data files.
5. RMAN recovers the data files with incremental backups and archived redo log files to a noncurrent point in time. RMAN must perform database point-in-time recovery, even when no explicit point in time is provided for duplication. Point-in-time recovery is required because the online redo log files in the source database are not backed up and cannot be applied to the duplicate database. The farthest point of recovery of the duplicate database is the most recent redo log file archived by the source database.

Understanding the RMAN Duplication Operation

When you execute the `DUPLICATE` command, RMAN performs the following operations:

6. Shuts down and restarts the auxiliary instance in `NOMOUNT` mode
7. Creates a new control file, which then creates and stores the new `DBID` in the data files
8. Opens the duplicate database with the `RESETLOGS` option
9. Creates the online redo log files for the duplicate database

Note: The database duplication process attempts to resume from the point-of-failure upon reexecution.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Understanding the RMAN Duplication Operation (continued)

6. RMAN shuts down and restarts the database instance in `NOMOUNT` mode.
7. RMAN creates a new control file, which then creates and stores the new, unique database identifier `DBID` in the data files of the duplicated database.
8. RMAN opens the duplicate database with the `RESETLOGS` option.
9. RMAN creates the online redo log files for the duplicate database.

Note: If the `DUPLICATE DATABASE` command fails, you can re-execute the `DUPLICATE DATABASE` command and the duplication process attempts to resume from the point-of-failure.

Specifying Options for the DUPLICATE Command

You can specify the following options with the DUPLICATE command:

Option	Purpose
SKIP READONLY	Excludes read-only tablespaces
SKIP TABLESPACE	Excludes named tablespaces
TABLESPACE	Includes named tablespaces
NOFILENAMECHECK	Prevents checking of file names
OPEN RESTRICTED	Enables RESTRICTED SESSION automatically

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Specifying Options for the DUPLICATE Command

Specify additional options when executing the DUPLICATE command as appropriate.

SKIP READONLY: Use to exclude read-only tablespace data files.

SKIP TABLESPACE: Use to exclude tablespaces from the target database. You cannot exclude the SYSTEM tablespace or tablespaces containing undo or rollback segments.

TABLESPACE: Use to include tablespaces from the target database.

NOFILENAMECHECK: Use to prevent RMAN from checking whether target database data files with the same name as duplicate database data files are in use. You must specify this option when the target database and duplicate database data files and redo log files use the same names. You would typically use this when you create a duplicate database on a host that has the same disk configuration, directory structure, and file names as the target database host. If you do not specify NOFILENAMECHECK in this situation, RMAN returns an error.

OPEN RESTRICTED: Use to enable RESTRICTED SESSION automatically after the database is opened.

Using Additional DUPLICATE Command Options

Option	Purpose
NOREDO	Signals RMAN that the application of redo logs should be suppressed during recovery Must be used with targetless DUPLICATE when target database is in NOARCHIVELOG mode at backup time Can also be used to explicitly state that no archived redo log files should be applied
UNDO TABLESPACE	Must be specified when target database is not open and there is no recovery catalog connection so that RMAN does not check the tablespace for SYS-owned objects

ORACLE

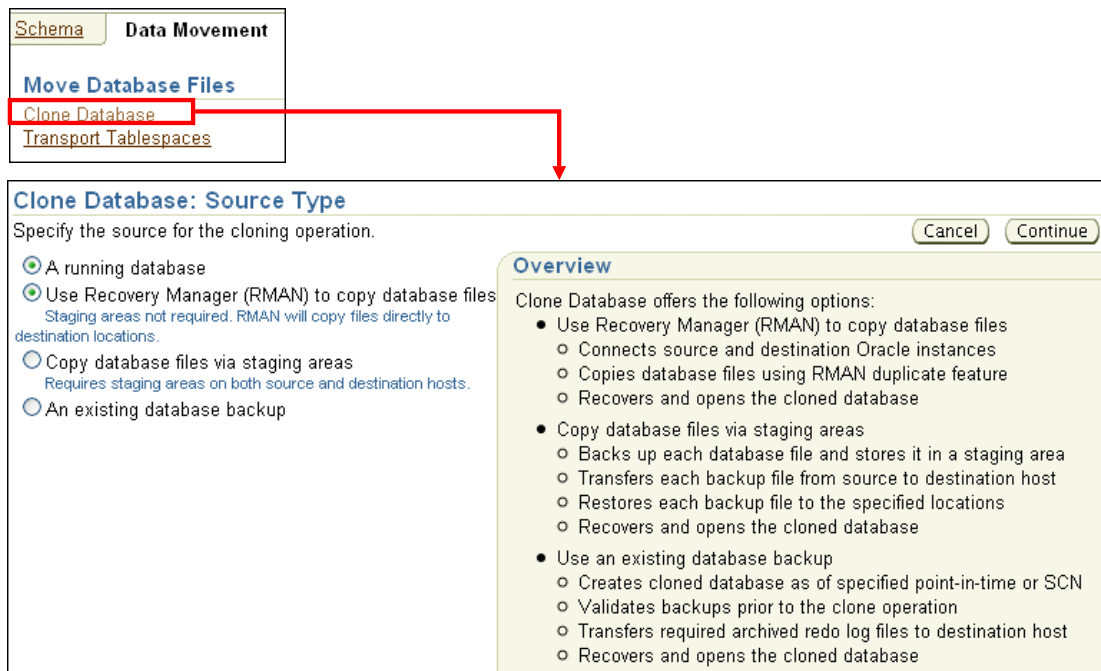
Copyright © 2009, Oracle. All rights reserved.

Using Additional DUPLICATE Command Options

The following additional options for the DUPLICATE command are introduced with Oracle Database 11g Release2:

- **NOREDO:** The NOREDO option is used to signal RMAN that redo logs should not be applied during the recovery phase of the duplication operation. This option should be specified when the database was in NOARCHIVELOG mode at the time of the backup or when the archived redo log files are not available for use during the duplication operation. This option is appropriate if a database that is currently in ARCHIVELOG mode is being duplicated to a point-in-time when it was in NOARCHIVELOG mode.
If you are performing a targetless DUPLICATE and the database is in NOARCHIVELOG mode, you must use this option to inform RMAN of the database mode. Without a connection to the target database, RMAN cannot determine the mode.
- **UNDO TABLESPACE:** RMAN checks that there are no objects belonging to the SYS user in any of the duplicated tablespaces during non-whole database duplication. The SYSTEM, SYSAUX, and undo segment tablespaces are excluded from this check. However, if the target database is not open and a recovery catalog is not being used during the duplication, RMAN cannot obtain the undo tablespace names. So you must use the UNDO TABLESPACE option to provide the names of undo segment tablespaces.

Using EM to Clone a Database



ORACLE

Copyright © 2009, Oracle. All rights reserved.

Using EM to Clone a Database

You can also use Enterprise Manager (EM) to create a duplicate (clone) database. From the EM home page, navigate to Data Movement > Clone database. The screenshot displays the Clone Database: Source Type page.

You can choose from the following venues for the clone operation:

- **Running instance:** You can specify a running instance to be cloned.
- **Staging area:** A disk area specified on the source and destination hosts. A backup is created and stored here, then put in the destination staging area, and read from on that destination host to create the clone database.
- **Existing backup:** If you already have a backup that reflects the database in the state you want it cloned, you can use that.

Quiz

Select all statements that are true about database duplication:

1. You can duplicate a database with or without connection to the auxiliary instance.
2. You can duplicate a database with or without connection to the recovery catalog.
3. You can duplicate a database with or without target connection.
4. You can duplicate a database only when you have RMAN backups.
5. You always have to manually re-create control files on the auxiliary instance.

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Answer: 2, 3

Summary

In this lesson, you should have learned how to:

- List the purposes of creating a duplicate database
- Choose a technique for duplicating a database
- Duplicate a database with RMAN
- Use an RMAN backup to duplicate a database
- Duplicate a database based on a running instance

ORACLE

Copyright © 2009, Oracle. All rights reserved.

Practice 20 Overview: Duplicating a Database

This practice covers cloning a database and using utilities to complete the setup of a functioning duplicated database.

ORACLE

Copyright © 2009, Oracle. All rights reserved.