
Oracle9i New Features for Administrators

Student Guide • Volume 1

D11318GC11
Edition 1.1
July 2001
D33452

ORACLE®

Authors

David Austin
Ric van Dyke
Jean-Francois Verrier
Michael Möller
Lex de Haan

Technical Contributors and Reviewers

Harald van Breederode
Bruce Ernst
Lothar Flatz
Scott Gossett
Arturo Gutierrez
Merrill Holt
Magnus Isaksson
Martin Jensen
Sushil Kumar
Stefan Lindblad
Russ Lowenthal
Patricia McElroy
Sujatha Muthulingam
Peter Sharman
Sabine Teuber

Publisher

May Lonn Villareal

Copyright © Oracle Corporation, 2001. All rights reserved.

This documentation contains proprietary information of Oracle Corporation. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with "Restricted Rights," as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

All references to Oracle and Oracle products are trademarks or registered trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

Contents

I Introduction

- Overview I-2
- Outline I-3
- Migration I-5
- iSQL*Plus I-6
- Oracle9i Sample Schemas I-7
- Five Sample Schemas I-8
- Optional Schedule I-9
- Student Preface I-10

1 Oracle Server Security

- Objectives 1-2
- Connecting as the DBA 1-3
- Example Connects 1-4
- Stricter Default Security 1-5
- Secure Application Role 1-7
- Example Application Role 1-8
- Global Application Context 1-9
- Managing Global Application Context 1-10
- Global Application Context Function 1-11
- Global Context Example 1-12
- Fine-Grained Access Control Enhancements 1-13
- Policy Groups 1-14
- Partitioned Fine-Grained Access Control 1-15
- Fine-Grained Audit 1-16
- Fine-Grained Auditing Implementation 1-17
- Fine-Grained Auditing Example 1-18
- FGA Event Handler 1-19
- Encryption Enhancements 1-20
- Oracle Label Security 1-21
- Oracle Login Server 1-22
- Oracle Enterprise Login Assistant 1-23
- Enterprise User Security Enhancements 1-24
- Summary 1-25
- Practice 1-1 Overview 1-26

2 General High Availability Technology

- Objectives 2-2
- Reducing Unplanned Downtime Overview 2-3
- Minimal I/O Recovery 2-4
- Fast-Start Time-Based Recovery Limit 2-7
- Fast-Start Time-Based Recovery Limit Overview 2-8
- Fast-Start Time-Based Recovery Limit 2-9
- Changes to Previous Parameters 2-12
- Changes to V\$INSTANCE_RECOVERY 2-13
- Oracle Flashback Overview 2-14
- Oracle Flashback 2-15

- Oracle Flashback and Oracle LogMiner 2-19
- Resumable Space Allocation Overview 2-20
- Life Cycle for Resumable Space Allocation 2-21
- Resumable Space Allocation Operations 2-23
- Enable Session Resumable Space Allocation 2-24
- DBMS_RESUMABLE Package 2-26
- AFTER SUSPEND System Event 2-28
- RESUMABLE System Privilege 2-30
- DBA_RESUMABLE Dictionary View 2-31
- Export/Import Enhancements (STATISTICS) 2-33
- Export/Import Enhancements (TABLESPACES Mode) 2-34
- Export/Import Enhancements (Resumable Space Allocation) 2-35
- Export/Import Enhancements (Flashback) 2-36
- Summary 2-37
- Practice 2-1 Overview 2-38
- Practice 2-2 Overview 2-39

3 Oracle9i LogMiner Enhancements

- Objectives 3-2
- LogMiner New Features 3-3
- DDL Statement Support 3-5
- Data Dictionary Access 3-7
- Dictionary Information in the Redo Logs 3-8
- Using an Online Data Dictionary 3-9
- DDL Tracking in the Dictionary 3-10
- Dictionary Staleness Detection 3-11
- Skip Past Log Corruptions 3-12
- Display Only Committed Transactions Information 3-13
- Primary Key Information 3-14
- LogMiner Restrictions 3-16
- LogMiner Views 3-17
- LogMiner Viewer 3-18
- Summary 3-25
- Practice 3-1 Overview 3-26

4 Backup and Recovery

- Objectives 4-2
- RMAN Manageability Enhancements 4-3
- Persistent Configuration Parameters 4-4
- Retention Policies 4-5
- CONFIGURE RETENTION POLICY 4-6
- Automatic Channel Allocation 4-7
- CONFIGURE CHANNEL 4-8
- CONFIGURE DEVICE TYPE ... PARALLELISM 4-11
- CONFIGURE DEFAULT DEVICE TYPE 4-12

- CONFIGURE BACKUP COPIES 4-13
- CONFIGURE EXCLUDE 4-14
- CONFIGURE SNAPSHOT CONTROLFILE and CONFIGURE AUXNAME 4-15
- CONFIGURE CONTROLFILE AUTOBACKUP 4-16
- Long Term Backups 4-18
- Mirrored Backups 4-19
- Backup File Optimization 4-20
- Restartable Backups 4-21
- Archive Log Backup 4-22
- Backupset Backup 4-23
- Restore File Optimization and Restartable Restore 4-24
- Recovery Manager Enterprise Manager Interface 4-25
- RMAN Reliability Enhancements 4-26
- Archive Log Failover and Automatic Log Switch 4-27
- Backup Piece Failover 4-28
- Block Media Recovery (BMR) 4-29
- MTTR Reduction 4-30
- Other BMR Benefits 4-31
- Recovery Manager Interface 4-32
- Trial Recovery 4-36
- Miscellaneous RMAN Enhancements 4-40
- REPORT OBSOLETE 4-41
- REPORT NEED BACKUP 4-43
- Command Unification 4-44
- LIST Command Improvements 4-45
- LIST Command New Syntax 4-46
- New SHOW Command 4-49
- CROSSCHECK Autolocate 4-51
- Other RMAN Enhancements 4-53
- Summary 4-54
- Practice 4-1 Overview 4-55

5 Oracle9i Data Guard

- Objectives 5-2
- Oracle9i Data Guard Overview 5-4
- Oracle9i Data Guard Architecture 5-6
- Data Guard Broker and Data Guard Manager 5-7
- Oracle9i Data Guard Broker 5-8
- Oracle9i Data Guard Manager 5-10
- No Data Loss and No Data Divergence Definitions 5-12
- Data Availability Modes in Data Guard 5-13
- Data Availability Mode Configuration Process 5-14
- Data Availability Mode Configuration Matrix 5-16

- Redo Log Reception Possibilities 5-17
- Creating Standby Redo Logs 5-18
- Setting a Failure Resolution Policy 5-19
- Finishing Managed Recovery 5-20
- Physical Standby Database Failover 5-21
- Physical Standby Database Graceful Switchover 5-22
- Database Switchover Steps 5-24
- Automatic Recovery of Log Gaps 5-27
- Automatic Recovery of Log Gaps Configuration 5-28
- Archive Gaps Monitoring 5-29
- Standby File Management 5-30
- Background Managed Recovery Mode 5-31
- Monitoring the Managed Recovery 5-32
- Updating the Standby at a Lag 5-33
- Parallel Recovery 5-34
- Miscellaneous Enhancements 5-35
- Summary 5-38

6 Database Resource Manager Enhancements

- Objectives 6-2
- Active Session Pool 6-3
- Active Session Pool Mechanism 6-4
- Active Session Pool Parameters 6-5
- Setting the Active Session Pool 6-6
- Maximum Estimated Execution Time 6-7
- Automatic Consumer Group Switching 6-8
- Undo Quota 6-9
- Changing Undo Quota 6-10
- Modified Views to Support Database Resource Manager Extensions 6-11
- Modified Catalog Tables and Views 6-12
- An Example Using Several Resource Allocation Methods 6-13
- Oracle Supplied Plans 6-15
- Summary 6-16

7 Online Operations

- Objectives 7-2
- Online Index Rebuild 7-3
- Index-Organized Table High Availability Enhancements 7-6
- Online Operations on IOTs Operations on Secondary Indexes 7-7
- Online Operations on IOTs Online Update of Logical ROWIDs 7-8
- Online Operations on IOTs Online Move 7-9
- Online Table Redefinition 7-10
- Online Table Redefinition Syntax 7-13

- Online Table Redefinition: Synchronization and Abort 7-15
- Online Table Redefinition: Example 7-16
- Online Table Redefinition Limitations 7-18
- Online Analyze Validate 7-20
- Quiesce Database Overview 7-21
- Quiesce Database Benefits 7-23
- Quiesce Database Syntax 7-24
- Quiesce Database Limitations 7-25
- Viewing the Quiesce State of an Instance 7-26
- Server Parameter File (SPFILE) 7-27
- What Is an SPFILE? 7-28
- Creating a SPFILE 7-29
- Viewing the Parameter Settings 7-30
- Changing Parameter Values Within a SPFILE 7-31
- STARTUP Command Behavior 7-32
- Exporting a SPFILE 7-33
- Example of an Exported SPFILE 7-34
- Migrating to a SPFILE 7-35
- Summary 7-36
- Practice 7-1 Overview 7-37
- Practice 7-2 Overview 7-38

8 Segment Management (Part I)

- Objectives 8-2
- Global Index Maintenance During Partition DDLs 8-3
- Benefits of Maintaining Global Indexes During DDL 8-4
- Maintaining Global Indexes During DDL 8-5
- Update or Rebuild Global Indexes? 8-8
- List Partitioning Overview and Benefits 8-9
- List Partitioning Example 8-10
- List Partitioning Pruning 8-11
- ALTER TABLE ADD PARTITION 8-12
- ALTER TABLE MERGE PARTITION 8-14
- ALTER TABLE MODIFY PARTITION ADD VALUES 8-16
- ALTER TABLE MODIFY PARTITION DROP VALUES 8-17
- ALTER TABLE SPLIT PARTITION 8-19
- List Partitioning Usage 8-21
- Metadata API 8-23
- Metadata API in Oracle9i 8-24
- Metadata API in Oracle9i Browsing Example 8-25
- Common Extraction, Transformation, and Loading (ETL) Process 8-26
- Pipelined Data Transformation in Oracle9i 8-27
- Overview of External Tables 8-28
- Applications of External Tables 8-29
- Example of Defining External Tables 8-30

Querying External Tables 8-31
Data Dictionary Information for External Tables 8-32
Summary 8-33
Practice 8-1 Overview 8-34
Practice 8-2 Overview 8-35

9 Segment Management (Part II)

Objectives 9-2
Automatic Segment-Space Management 9-3
Automatic Segment-Space Management at Work 9-4
Creating an Automatic Space Management Segment 9-6
Creating an Automatic Space Management Segment: Example 9-7
Modifications to the DBMS_SPACE Package 9-8
Block Space Management 9-9
DBMS_REPAIR Package 9-10
DBMS_REPAIR and PCTFREE Implementation 9-11
Modifications to Dictionary Views 9-12
Compatibility and Migration 9-13
What Is a Bitmap Join Index? 9-15
Advantages and Disadvantages 9-16
Example With Three Tables 9-17
Data Dictionary and Bitmap Join Indexes 9-18
Bitmap Join Indexes Restrictions 9-19
Summary 9-20
Practice 9-1 Overview 9-21

10 Performance Improvements

Objectives 10-2
Identifying Unused Indexes 10-3
Enabling and Disabling Monitoring Index Usage 10-4
V\$OBJECT_USAGE View 10-5
Skip Scanning of Indexes 10-7
Skip Scanning Example 10-8
Cursor Sharing Enhancements 10-17
CURSOR_SHARING Parameter 10-19
Cached Execution Plans 10-20
New View to Support Cached Execution Plans 10-21
New PLAN_HASH_VALUE Column in V\$SQL 10-22
New First Rows Optimization 10-23
New Gathering Statistic Estimates 10-24
New GATHER AUTO Option 10-26
Optimizer Cost Model Enhancements 10-27
Gathering System Statistics 10-29
Gathering System Statistics Example 10-30

Summary 10-32
Practice 10-1 Overview 10-33
Practice 10-2 Overview 10-34

11 Scalable Session State Management

Objectives 11-2
Oracle Shared Server Improvements 11-3
Connection Establishment: Direct Handoff 11-4
Common Event Model for Database and Network 11-5
Performance Manager Monitoring 11-6
External Procedure Agent Enhancements 11-7
Dedicated External Procedure Agents 11-8
Libraries for External Procedures 11-9
Example 11-10
Multithreaded HS Agent Architecture 11-12
Multithreaded HS Agent Threads 11-13
HS Agent Initialization Parameters 11-14
OCI Connection Pooling: Features 11-15
Usage Model 11-16
Steps Used for OCI Connection Pooling 11-17
Core Library Improvements 11-18
Summary 11-19

12 Real Application Clusters

Objectives 12-2
Real Application Clusters Overview 12-3
Benefits of Real Application Clusters 12-4
Global Cache Service 12-5
Dynamic Resource Remastering 12-6
Global Cache Service Resource Modes 12-7
Global Cache Service Resource Roles 12-8
Block Transfers 12-9
Block Transfer Examples 12-10
Consistent Read Block Transfer 12-12
Cache Fusion Block Transfer 12-13
Benefits of Cache Fusion 12-15
High Availability Features 12-16
Real Application Clusters Guard 12-17
Real Application Clusters Guard Architecture 12-18
Monitors and Failover 12-19
Shared Initialization Parameter File 12-20
Shared Server-Side Parameter File 12-21
SRVCTL Commands 12-22
Real Application Clusters and OEM 12-23

- OEM Instance Management Provisions 12-24
- Background Processes 12-25
- Initialization Parameters 12-26
- Obsolete Global Cache Parameters 12-28
- Summary 12-29

13 File Management

- Objectives 13-2
- Oracle-Managed Files Overview 13-3
- Who Can Use Oracle-Managed Files? 13-4
- New Dynamic Initialization Parameters 13-5
- OMF Example 13-6
- OMF File Names Structure 13-7
- Managing OMF Control Files: Database Creation 13-9
- Impact of OMF on CREATE CONTROLFILE Command 13-10
- Managing OMF Redo Log Files 13-12
- Managing OMF Tablespaces 13-13
- OMF Examples 13-14
- OMF and Standby Databases 13-15
- Automatically Drop Non-OMF Data Files 13-16
- Default Temporary Tablespace 13-17
- Create Default Temporary Tablespace at Database Creation Time 13-19
- Alter Default Temporary Tablespace 13-20
- Restrictions on Default Temporary Tablespace 13-21
- Summary 13-22
- Practice 13-1 Overview 13-23
- Practice 13-2 Overview 13-24

14 Tablespace Management

- Objectives 14-2
- Automatic Undo Management 14-3
- Automatic Undo Management Concepts 14-4
- Dynamic Extents Transfer 14-6
- Specifying the Mode for Undo Space Management 14-7
- Creating an Undo Tablespace at Database Creation Time 14-9
- Creating an Undo Tablespace After Database Creation 14-10
- Altering an Undo Tablespace 14-11
- Dropping an Undo Tablespace 14-12
- Switching Undo Tablespaces 14-13
- Changing the Retention Period for Undo Information 14-15
- Data Dictionary View to Support Automatic Undo Management 14-17
- New Data Dictionary View to Support Automatic Undo Management 14-18
- New Parameters to Support Automatic Undo Management: Summary 14-20
- Multiple Block Size Support 14-21

- Standard Block Size 14-22
- Nonstandard Block Sizes 14-24
- Creating a Nonstandard Block Size Tablespace 14-25
- Multiple Block Sizing Rules 14-26
- Summary 14-27
- Practice 14-1 Overview 14-28
- Practice 14-2 Overview 14-29

15 Memory Management

- Objectives 15-2
- Automated SQL Execution Memory Management 15-3
- PGA Memory Management 15-4
- Enabling Automated SQL Execution Memory Management 15-5
- New Statistics and Columns 15-7
- Example 15-9
- New Views to Support SQL Execution Memory Management 15-10
- Dynamic SGA 15-12
- Unit of Allocation 15-13
- Growing a Component's SGA Memory Area 15-15
- Dynamic Shared Pool 15-16
- Dynamic Buffer Cache 15-17
- New Buffer Cache Parameters 15-18
- Deprecated Buffer Cache Parameters 15-19
- Example Buffer Caches Setup 15-20
- Dynamic Buffer Cache Advisory Parameter 15-21
- New View to Support Buffer Cache Advisory 15-22
- V\$DB_CACHE_ADVICE Example 15-23
- Summary 15-24

16 Enterprise Manager Enhancements

- Objectives 16-2
- New Console Look and Feel 16-3
- Launching Enterprise Manager Console 16-4
- Console Launched Standalone 16-5
- Connections Using Management Server 16-6
- Standalone Connection Benefits 16-7
- Standalone Connection Restrictions 16-8
- Standalone Repository 16-9
- Enterprise Manager Support for Oracle9i Database Features 16-10
- Creating the SPFILE 16-11
- Creating a PFILE from SPFILE 16-12
- Changing Parameters 16-13
- Startup Using the SPFILE 16-14
- Undo Tablespace Support 16-15
- Undo Tab 16-16
- Buffer Cache Size Advice View 16-17
- Creating Default Temporary Tablespace 16-18

- Mean Time to Recovery 16-19
- Backup and Recovery Enhancements 16-20
- Advanced Queuing 16-21
- HTML Database Reports 16-22
 - Database Configuration Report 16-23
- User-Defined Events 16-24
- User Defined Event Tests 16-25
- Event Handler 16-27
- Summary 16-28

17 SQL Enhancements

- Objectives 17-2
- SQL:1999 Enhancements Overview 17-3
- SQL:1999 Joins 17-4
 - Cross Joins 17-5
 - Natural Joins 17-6
 - Natural Join Example 17-7
 - Equijoins and the USING Clause 17-8
 - USING Clause Example 17-9
 - Join Predicates and the ON Clause 17-10
 - Three-Way Joins with the ON Clause 17-11
 - Outer Joins 17-12
 - Outer Join Example 17-13
- CASE Expression Enhancements 17-14
 - Simple CASE Expressions 17-15
 - Searched CASE Expressions 17-16
- NULLIF and COALESCE 17-17
- Scalar Subqueries 17-18
 - Scalar Subquery Example 17-19
- Explicit Defaults 17-20
- The MERGE Statement 17-21
- Analytical Function Enhancements 17-22
 - WIDTH_BUCKET Function 17-24
 - WIDTH_BUCKET Example 17-25
- Grouping Sets 17-26
- Composite Columns 17-28
- Concatenated Groupings 17-30
- WITH Clause 17-31
 - WITH Clause Example 17-32
- Constraint Enhancements 17-33
 - Explicit Index Control 17-34
- Less Foreign Key Locking Overhead 17-35
- Cached Primary Key Look Up 17-36
- Constraints on Views 17-37
- View Constraint Types 17-38

- Create Constrained Views 17-39
- Constrained View Maintenance 17-40
- Index Scans and Function-Based Indexes 17-41
- SELECT ... FOR UPDATE WAIT 17-42
- Multitable INSERT Statement 17-43
- Multitable INSERT Syntax 17-44
- LONG to LOB Migration 17-45
- PL/SQL Support for LOB Migration 17-46
- Restrictions on LOB Migration 17-47
- Common SQL Parser 17-48
- Native PL/SQL Execution 17-49
- Summary 17-50
- Practice 17-1 Overview 17-51
- Practice 17-2 Overview 17-52

18 Globalization Support

- Objectives 18-2
- Globalization and NLS 18-3
- New Time and Interval Data Types 18-4
- New Time and Interval Data Type Example 18-5
- TIMESTAMP Literals 18-6
- INTERVAL Literals 18-7
- Formatting NLS Variables 18-8
- Using Time Zones 18-9
- Application or Server Time Zone Handling 18-10
- Datetime/Interval Arithmetic 18-11
- Datetime Functions 18-12
- Datetime Conversion Functions 18-13
- Datetime Extract Function 18-14
- Daylight Saving Time 18-15
- Unicode 18-16
- Unicode Encodings 18-17
- Unicode Character Forms 18-18
- Enhanced Unicode Support 18-19
- Migration and Unicode Issues 18-21
- New Unicode Character Sets 18-22
- Choosing a Unicode Solution Scenario Unicode Database 18-23
- Choosing a Unicode Solution Scenario Unicode Data Type 18-24
- More Character Sets, Languages, and Territories 18-25
- Enhanced Linguistic Sorting 18-26
- New Linguistic Sorts 18-27
- Byte and Character Semantics 18-29
- Implicit Type Conversion for NCHAR Data Type 18-31
- SQL*Loader Unicode Support 18-32
- SQL*Loader for Unicode Sample Control File 18-33

- Character Set Scanner 18-34
- Common Character Conversion Problems 18-35
- Character Set Scanner Operation 18-37
- Character Set Scanner Command 18-38
- Character Set Scanner Output 18-39
- Character Conversion 18-42
- Oracle Locale Builder 18-44
- Locale Builder Examples 18-45
- New or Modified Globalization Settings 18-46
- Summary 18-47
- Practice 18-1 Overview 18-48

19 Database Workspace Management

- Objectives 19-2
- What Is a Database Workspace? 19-3
- How Does Workspace Manager Work 19-4
- Workspace Manager Administrator Role 19-5
- Version Enable a Table 19-6
- Changes Due to Versioning 19-7
- The History Option 19-9
- Guidelines for Tables Participating in a Workspace 19-10
- Disabling Workspace Participation for a Table 19-12
- Workspace Savepoints 19-13
- Create a Workspace 19-14
- Assign Workspace: Associate a User 19-15
- Rows in the Table 19-16
- Assign Workspace: Grant Privileges 19-17
- Assign Workspace: Set Locks 19-19
- Freeze a Workspace 19-20
- Roll Back a Workspace 19-21
- Refresh a Workspace 19-22
- Resolve Workspace Conflicts 19-23
- Conflict Resolution Example: Check for Existence of Conflicts 19-24
- Merge a Workspace 19-25
- Import and Export Considerations 19-26
- Enterprise Manager Interface 19-27
- Workspace Metadata Views 19-29
- Summary 19-31
- Practice 19-1 Overview 19-32

20 Advanced Replication

- Objectives 20-2
- Extended Availability of Replication Environment 20-3
- Add New Master Site Without Quiescing 20-4
- SPECIFY_NEW_MASTERS 20-6

ADD_NEW_MASTERS 20-7
RESUME_PROPAGATION_TO_MDEF 20-9
Perform Import or Change-Based Recovery 20-10
PREPARE_INSTANTIATED_MASTER 20-11
When to Add New Masters Without Quiescing 20-12
Restrictions on Adding New Masters Without Quiescing 20-13
Row-Level System Change Numbers 20-14
Parallel Propagation and Row-Level SCNs 20-16
Constraint SCNs 20-17
Materialized View Fast Refresh Abilities 20-18
Explain Materialized View 20-20
MV_CAPABILITIES_TABLE 20-21
Multitier Materialized Views 20-22
Replication of Objects 20-24
Replicating Objects with Materialized Views 20-25
Monitoring and Managing Replication Environments 20-26
Job Queue Changes 20-27
LONG to LOB Migration 20-28
Changes for Related Oracle9i Features 20-29
Installation and Upgrade 20-30
Replication Support for Unicode 20-31
Summary 20-32

A Practices

B Solutions

Introduction to Oracle9i: New Features for Administrators

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

Overview

- **This course concentrates on Oracle9i new features applicable to database administration**
- **Previous experience with Oracle databases is required for a full understanding of many new features, particularly Oracle8 and Oracle8i**
- **Hands-on practices emphasize functionality rather than testing knowledge**

ORACLE

I-2

Copyright © Oracle Corporation, 2001. All rights reserved.

Overview

This course is designed to introduce you to the new features of Oracle9i that are applicable to the work normally performed by database administrators and related personnel. The course does not attempt to provide every detail about a feature or cover aspects of a feature that were available in previous releases, other than when defining the context for a new feature or comparing past behavior with current behavior. Consequently, the course will be most useful to you if you have already administered other versions of Oracle databases, particularly Oracle8 and Oracle8i. Even with this background, you should not expect to be able to implement all of the features discussed in the course without supplemental reading, particularly the Oracle9i documentation.

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

Outline

This course is organized around important functions for a database administrator:

- **Security**
 - Key to maintaining database integrity
 - Important changes that impact common database administrative tasks
- **Availability**
- **Configuration and tuning**
- **Manageability**
- **Development platform**

ORACLE

I-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Outline

With the assumption that you are a database administrator, the course lessons are organized by topics reflecting the key functions associated with your work.

The first topic, Security, is covered in a single lesson at the very beginning of the course. There are two reasons for this lesson appearing first:

- Most security administrators will tell you that having an insecure database is equivalent to not having a database at all.
- There are some changes in the way you access the database to perform startup and shutdown operations, along with other administrative tasks, that might be different from how you are used to working. You will need this information to complete some of the hands-on practices later in the course.

Outline

- **Availability: covers features to reduce database down-time and data loss**
- **Configuration and tuning: addresses setup and performance issues**
- **Manageability: covers tools and features that reduce the work required to keep your database running**
- **Development platform topics: covers features that could impact database behavior if used by application developers**

ORACLE

I-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Outline (continued)

Besides security, the other topics covered in the course are:

- Availability, which addresses features to reduce interruptions in database service, updates to tools such as Export/Import, LogMiner, and Recovery Manager, as well as new backup and recovery options
- Configuration and tuning, covering topics such as:
 - New online operations, monitoring options, and options to scale applications
 - Automatic segment free space management
 - Optimizer performance improvements
 - Server-side parameter files
 - Enhancements to tools including Database Resource Manager, DBMS_SPACE, DBMS_REPAIR, and Real Application Clusters
- Manageability includes a wide variety of topics covering physical storage management, memory management and configuration, rollback and temporary segment management, and the use of multiple block sizes in a single database. This section also covers changes to management features of Oracle Enterprise Manager.
- Development platform topics address features that might impact your database if your applications are rewritten to take advantage of them. Some of the new features may also help you directly, such as extensions to a number of SQL commands.

Oracle9i: New Features for Administrators I-4

Migration

Migration paths from older releases to Oracle9i:

- **For Version 6 and Version 7**
 1. **Migrate to Oracle8 or Oracle8i**
 2. **Upgrade to Oracle9i**
- **For Releases 8.0.1 to 8.0.5, upgrade to 8.0.6 or higher**
- **For Version 8.0.6 or 8.1 and higher, upgrade using:**
 - **Oracle Data Migration Assistant**
 - **Manual upgrade**

ORACLE

I-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Migration

To upgrade your database to release Oracle9i, your current database must be at release 8.0.6 or higher. For older releases, you must first migrate to release Oracle8 or Oracle8i. The migration manual for each of these releases describes the migration procedures for that release. This information is also repeated in the *Oracle9i: Database Migration* manual.

To upgrade your database to Oracle9i from release Oracle8 or Oracle8i, use the Oracle Data Migration Assistant or perform the upgrade manually.

The Oracle Data Migration Assistant provides an automated upgrade of your database using a graphical user interface (GUI). In addition, the Oracle Data Migration Assistant includes extensive online help. The Oracle Data Migration Assistant runs the appropriate upgrade script for your current release, deletes any obsolete initialization parameters from your initialization parameter file, and optionally configures your `listener.ora` file.

On the other hand, you lose some flexibility and control by using the Oracle Data Migration Assistant. If you want complete control over the upgrade process, especially with regard to setting initialization parameters, then you might want to perform the upgrade manually. If your system has Oracle Parallel Server installed, the Oracle Data Migration Assistant cannot upgrade it and you must upgrade your database manually.

Both upgrade methods are fully explained in the *Oracle9i: Database Migration* manual.

iSQL*Plus

The screenshot shows the iSQL*Plus web interface. At the top, there's a title bar 'iSQL*Plus Release 9.0.1'. Below it, the Oracle logo and 'iSQL*Plus' text are on the left, and 'Password', 'Log Out', and 'Help' links are on the right. A 'Script Location' field with a 'Browse...' button and a 'Load Script' button are present. Below this is a text area for 'Enter statements:' containing the SQL query: `select * from employees where salary > 13000`. Below the text area are buttons for 'Execute', 'Output' (set to 'Work Screen'), 'Clear Screen', and 'Save Script'. At the bottom, a table displays the results of the query.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000

iSQL*Plus

iSQL*Plus is a browser-based implementation of SQL*Plus. The three tiers, client, middle, and server, may be on the same machine, or on separate machines.

iSQL*Plus User Interface (Client Tier)

The iSQL*Plus user interface runs in a web browser connected to the Internet or your intranet. You only need to know the URL of the Oracle HTTP Server to access Oracle9i, or any Oracle database. There is no installation or configuration required for the iSQL*Plus user interface other than a web browser.

Oracle HTTP Server (Middle Tier)

The iSQL*Plus Server is installed as part of the Oracle HTTP Server. The iSQL*Plus Server enables communication and authentication between the iSQL*Plus user interface and the RDBMS.

There are two modes to connect to iSQL*Plus:

- As a standard user
- As a privileged user (AS SYSDBA or SYSOPER)

You can select the connection mode from the Privilege dropdown on the Log In screen.

For more information about iSQL*Plus, refer to the *iSQL*Plus User's Guide and Reference* or visit OTN at http://technet.oracle.com/tech/sql_plus/

Oracle9i Sample Schemas

- **One set of database objects for demos, documentation examples, and training material; grouped into schemas of staggered complexity**
- **Spend more time learning and less time understanding the example environment**
- **Oracle9i Universal Installer, DBCA**
- **Design principles:**
 - **Simplicity and ease of use**
 - **Fundamentals coverage**
 - **Extensibility**
 - **Relevance**

ORACLE

I-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits

Continuity of Context

When encountering the same set of tables everywhere, you spend less time with the schema and more time understanding the technical concepts.

Usability

You can use these schemas in the seed database to run examples that are shown in Oracle documentation and training materials. This first-hand access to examples facilitates both conceptual understanding and application development.

Quality

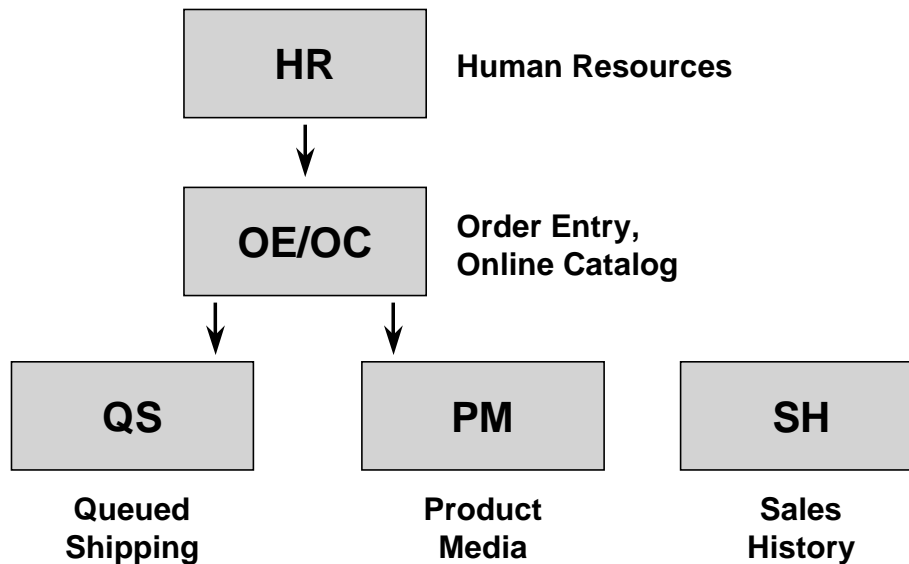
Through central maintenance and testing, the quality of Oracle documentation and training materials will be enhanced.

Design Principles

The base schemas and the extensions should bring to the foreground the functionality that customers typically use. Only the most commonly used database objects are built automatically in the schemas, and the entire set of schemas provides a foundation upon which one can expand to illustrate additional functionality.

The Oracle9i Sample Schemas are designed to be applicable to e-business and other significant industry trends (for example, XML). When this goal conflicts with the goal of simplicity, schema extensions are used to showcase the trends in focus.

Five Sample Schemas



ORACLE

I-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Five Sample Schemas

The new Oracle9i Sample Schemas are a set of interlinked schemas. This set of schemas is aimed at providing a layered approach to complexity:

Human Resources (HR)

A simple schema for introducing basic topics. An extension to this schema supports Oracle Internet Directory demos.

Order Entry (OE)

This schema deals with matters of intermediate complexity. A multitude of data types is available in this schema.

Online Catalog (OC)

This subschema is a collection of object-relational database objects built inside the OE schema.

Product Media (PM)

A schema dedicated to multimedia data types.

Queued Shipping (QS)

A set of schemas gathered under the main schema name to demonstrate Oracle Advanced Queuing capabilities.

Sales History (SH)

A schema designed to allow for demos with larger amounts of data. An extension to this schema provides support for advanced analytic processing.

Optional Schedule

Topic	Lessons	Day
Security	1	1
Availability	2 - 4	1
	5	2
Configuration and tuning	6 - 7	2
	8 - 12	3
Manageability	13 - 16	4
Development platform	17	4
	18 - 20	5

ORACLE

I-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Optional Schedule

The lessons in this guide are arranged in the order you will probably learn them in class. The lessons are grouped into the topic areas defined previously, but they are also organized by other criteria including the following:

- Introduce a feature in an early lesson before referencing that feature in later lessons.
- Alternate difficult and easy topics to facilitate learning.
- Distribute lessons with hands-on practices throughout the course to provide regular opportunities for students to explore what they are learning.

If your instructor teaches the class in the order the lessons are printed in this guide, the class should run approximately as shown in the schedule. Your instructor may vary the order of the lessons, however, for a number of valid reasons. These include:

- Customizing material for a specific audience
- Covering a topic in a single day rather than splitting the material across two days
- Maximizing the use of course resources such as hardware and software

Student Preface

Even if topics do not interest you now, your job functions may change due to:

- **New products or methods**
- **Changes in your employment**
- **Changes in the corporate structure**
- **Enhancements in applications**

ORACLE

I-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Student Preface

As you progress through this course, you may encounter a number of topics that hold no interest for you at the present time. However, you should remember that information technology is an industry in which about 50 per cent of your skills and knowledge today will be obsolete a year or so from now. Features that are not of current interest could become valuable to you for a number of reasons:

- Your company introduces a new product or business method that requires new database functionality.
- You choose to advance your career by moving to a different group or company which requires you to use unfamiliar features.
- Your company business changes due to a merger or acquisition.
- A maintenance window for an old application allows the developers to incorporate features that were not available when the application was first written.

1

Oracle Server Security

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Adapt to the new privileged connection options**
- **List the new security features and their application:**
 - **Secure Application Role**
 - **Global Context**
 - **Partitioned Fine Grain Access Control**
 - **Fine Grained Auditing**
- **Summary of optional security products**

ORACLE

Connecting as the DBA

- **CONNECT INTERNAL no longer possible**
 - Announced several versions earlier
 - Use the **AS SYSDBA** method instead
- **SVRMGR removed**
 - Too little difference between **SVRMGR** and **SQL*Plus**
 - **SVRMGR** special commands available in **SQL*Plus** since **Oracle8i**

ORACLE

1-3

Copyright © Oracle Corporation, 2001. All rights reserved.

De-support of **CONNECT INTERNAL** and Server Manager

For the past few releases, Oracle Corporation has stated that the **CONNECT INTERNAL** syntax to authenticate a database administrative connection was provided only for backwards compatibility. With the Oracle8i release, Server Manager was deprecated because all database management tools were available through SQL*Plus.

With the Oracle9i release, the **CONNECT INTERNAL** syntax and the Server Manager tool are discontinued. You can use a password file with **SYSOPER** or **SYSDBA** privileges enabled, or enable Operating System Authentication, and use the **OSOPER** and **OSDBA** roles. Use SQL*Plus for database **STARTUP**, **SHUTDOWN**, and related activities. These features are not new to Oracle9i.

Note: Connections using the **SYSOPER** or **SYSDBA** privileges are audited automatically just as **INTERNAL** connections were in earlier releases.

Migration of Scripts

Scripts that use **svrmgrl** need to be edited to use **sqlplus** instead.

The **charwidth** command is not available in SQL*Plus, because it has more flexible formatting with the **column** command. Long output lines wrap differently in SQL*Plus.

Example Connects

- Close equivalent to internal

```
SQL> connect / as sysdba
```

- With password file on remote system

```
SQL> connect system/manager@remote as sysdba
```

ORACLE

1-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Example Connects

```
$ sqlplus
```

```
SQL*Plus: Release 9.0.1.0.0 - Production on Mon Jul 02 06:17:41 2001
```

```
(c) Copyright 2000 Oracle Corporation. All rights reserved.
```

```
Enter user-name: system/manager
```

```
Connected to an idle instance.
```

```
SQL> connect internal -- fails, discontinued command
```

```
Enter password: oracle
```

```
ERROR:
```

```
ORA-09275: CONNECT INTERNAL is not a valid DBA connection
```

```
Warning: You are no longer connected to ORACLE.
```

```
SQL> connect sys/change_on_install
```

```
ERROR: -- fails, cannot verify username
```

```
ORA-01034: ORACLE not available -- when database is down
```

```
SQL> connect sys/change_on_install as sysdba
```

```
Connected to an idle instance. -- user has SYSDBA privilege and
```

```
-- is in the password file
```

```
SQL> connect / as sysdba
```

```
-- Operating system authenticated, using OSDBA
```

```
Connected to an idle instance. -- (close equivalent to "internal")
```

Note: These features have been available in earlier releases.

Stricter Default Security

- **Database Creation Assistant (DBCA) created databases have:**
 - **Locked accounts**
 - **No default passwords**
- **O7_DICTIONARY_ACCESSIBILITY defaults to FALSE**
 - **Only SYSDBA privileged users can see the data dictionary.**
 - **Previous default TRUE allowed data dictionary access to users with SELECT ANY TABLE.**

ORACLE

1-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Locked Accounts

On a new database, created by the Database Creation Assistant (DBCA), most of the standard accounts, such as CTXSYS and MDSYS are locked and have the password expired. The exceptions include SYS, SYSTEM, and SCOTT.

When unlocking these accounts, two steps are required:

- `SQL> ALTER USER hr ACCOUNT UNLOCK;`
- Define a new password, either by attempting a login with any password, or by using `SQL> ALTER USER hr IDENTIFIED BY <new_password>;`

No Default Password

If you create the database with the Database Creation Assistant (DBCA), it will not create the system accounts with a default defined password; you must enter a password.

If you create the database manually with `CREATE DATABASE`, the passwords for SYS and SYSTEM have the same default as before.

O7_DICTIONARY_ACCESSIBILITY

The default of this parameter prior to Oracle9i was TRUE. This enabled users with SELECT ANY to read the Data Dictionary (objects owned by SYS). The FALSE setting requires login with AS SYSDBA to read the data dictionary, or to be given explicit object grants.

Disable Some PUBLIC Grants

You should consider revoking the EXECUTION grant given to PUBLIC on some packages, and grant this to the few who may need execution grant on them. While the packages are not a direct security risk, they give access to some features which could be used by a malicious user to gain access elsewhere.

Secure Application Role

- **Solves the problem of preventing unauthorized access to data through other client programs**
- **Better than previous mechanism with “hidden password”**
- **Enabling a role is checked through a package, not a password**
- **Uses the same `SYS_CONTEXT` mechanism as Virtual Private Database**

ORACLE

1-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Enhanced Virtual Private Database

The concept of Application Context was introduced in Oracle8i, where it was used with Fine Grain Access Control (FGAC) to make a Virtual Private Database (VPD). Application Context can now be attached to a `ROLE`, thus enhancing the implementation of VPD.

To enable an application role, the associated procedure must be called, which controls whether the role is enabled or not. This can make a number of additional checks, using `SYS_CONTEXT('USERENV' , nnn)` to check for authenticity, where the `nnn` could be, for example, `IP_ADDRESS` or `PROXY_ACCOUNT`.

The method used prior to Oracle9i consisted of using password authentication of the role, and embedding and hiding the password in the application. The weak point in that method was that if the password was known, any application could access the data.

Example Application Role

```
CREATE ROLE salesuser  
IDENTIFIED USING sh.sales_chk;
```

```
CREATE OR REPLACE PROCEDURE sales_chk  
AUTHID CURRENT_USER IS  
ipchk STRING(30);  
BEGIN      /* Only certain IP addresses allowed */  
  SELECT SYS_CONTEXT('USERENV','IP_ADDRESS')  
  INTO ipchk FROM DUAL;  
  IF SUBSTR(ipchk,1,4) != '192.'  
  THEN RETURN; END IF;          /* Fail silently */  
  DBMS_SESSION.SET_ROLE('SALESUSER'); /* Enable */  
END;  
/
```

ORACLE

1-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Example Application Role

```
CREATE ROLE <rolename> IDENTIFIED USING <package name> ;
```

The example will create the role SALESUSER where access control is handled by the procedure SALES_CHK. The procedure does not have to exist for the CREATE ROLE command to succeed. The same procedure can be specified with several roles, for example, SALES_CHK could enable the role SALESUSER or SALESMGR depending on whether the connection, user, or program is appropriate.

The example procedure only allows access to certain IP addresses.

To enable the role, the user calls the procedure SALES_CHK. Any other attempt to enable the role will fail.

Note: The inverted logic of the code means that non-IP addresses will also be able to enable the role. In real use, you have to check for NULL returns from SYS_CONTEXT.

Global Application Context

- **A context can now be global and thus shared.**
- **The Global Application Context is:**
 - **Cheaper in resources than individual session context**
 - **Well suited to web based applications using Virtual Private Database**
 - **Still able to verify access rights by way of an *identifier***
 - **Well suited with connection multiplexing**

ORACLE

1-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Uses of Global Application Context

Oracle9i Virtual Private Database (VPD) capabilities facilitate connection pooling by allowing multiple connections to access one or more global application contexts, instead of setting up an application context for each user session. Global application contexts provide additional flexibility for web-based applications to use. Performance is enhanced through reuse of common application contexts among multiple sessions, instead of setting up per-session application contexts.

Application user proxy authentication can be used with global application context for additional flexibility and high performance in building eBusiness applications. This provides performance improvements through session reuse, and through accessing global application contexts set up once, instead of having to initialize application contexts for each session individually.

In order to decide the driving context for the current session to enforce fine-grain access control, the middle tier must set the application context for the session. The Global Context allows the middle tier to store the various application context definitions in a central place in the instance (SGA) and apply or assign the context to a user session at session creation time. This then becomes that session's driving context. This will greatly reduce the user session (application context setup) setup time in an application connection pooling environment.

Managing Global Application Context

A set of interfaces has been added to the **DBMS_SESSION** package to manage application context in client sessions:

- **SET_CONTEXT**
- **CLEAR_CONTEXT**
- **SET_IDENTIFIER**
- **CLEAR_IDENTIFIER**

ORACLE

1-10

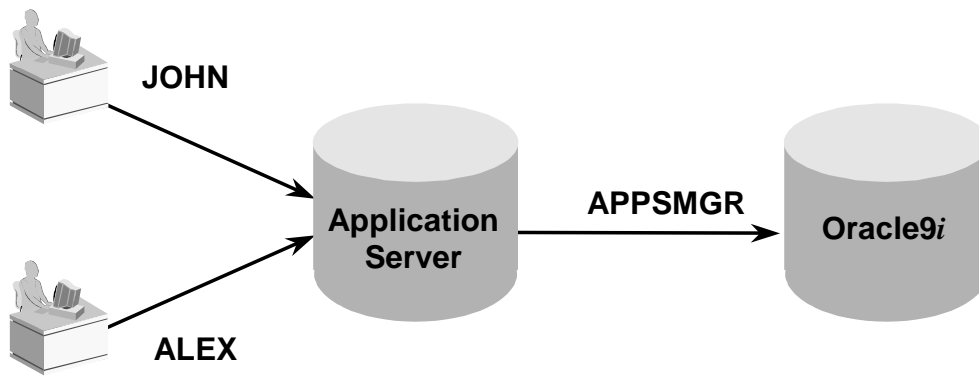
Copyright © Oracle Corporation, 2001. All rights reserved.

Managing Global Application Context

To support the application managed session pooling by middle tier applications, the **DBMS_SESSION** interface for managing application context is also enhanced to add a client identifier for each application context. In this way, application context can be managed globally while each client sees only their assigned application context.

The middle tier application server can use **SET_CONTEXT** to set application context for a specific client ID. Then, when assigning a database connection to process the client request, the application server needs to issue a **SET_IDENTIFIER** to denote the ID of the application session. From then on, every time the client invokes **SYS_CONTEXT**, only the context that was associated with the set identifier is returned.

Global Application Context Function



ORACLE

1-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Global Application Context Applied

The administrator creates the global context namespace by issuing:

```
SQL> CREATE CONTEXT webhr USING hr.init ACCESSED GLOBALLY;
```

The Application Server starts up and establishes multiple connections to the database as user HR.

For users, the following occurs:

User JOHN logs on to the application server. This is not a database username. The Application Server authenticates JOHN into the application, and assigns a temporary session ID for this connection, 4523, based on a unique application session attribute. This session ID is returned to John's browser as part of a cookie or maintained by the application server.

The application server initializes application context for this client calling `hr.init` package, which executes:

```
DBMS_SESSION.SET_CONTEXT('webhr','id','JOHN','HR',4523);  
DBMS_SESSION.SET_CONTEXT('webhr','dep','sales','HR',4523);
```

User ALEX logs in to the application server, and gets the number 8741. The application code issues similar `SET_CONTEXT` calls:

```
DBMS_SESSION.SET_CONTEXT('webhr','id','ALEX','HR',8741);  
DBMS_SESSION.SET_CONTEXT('webhr','dep','sales','HR',8741);
```

Global Context Example

```
CREATE CONTEXT webhr USING hr.init ACCESSED GLOBALLY;
```

```
DBMS_SESSION.SET_CONTEXT('webhr','id','JOHN','HR',4523);
```

```
DBMS_SESSION.SET_IDENTIFIER(4523);
```

...

SYS_CONTEXT calls are in John's context

...

```
DBMS_SESSION.CLEAR_IDENTIFIER(4523);
```

ORACLE

1-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Global Application Context Applied (continued)

When user JOHN uses the application server to actually access data, the application server picks any session it has logged in with APPSMGR and on it, executes:

```
DBMS_SESSION.SET_IDENTIFIER(4523);
```

All SYS_CONTEXT calls within this database session will only return application context values belonging to the client session only. For example, SYS_CONTEXT('webhr','id') will return 'JOHN'.

Similarly when ALEX uses the system a session connection is picked and

```
DBMS_SESSION.SET_IDENTIFIER(8741);
```

is issued by the application code. The session context status assumes Alex's state, irrespective of which session issued the SET_CONTEXT calls.

Finally, old identifiers can be cleared when users leave the application.

Note: It is assumed that data access is managed with Virtual Private Database. Establishing the context does not automatically limit data access.

Fine-Grained Access Control Enhancements

- **Allow several access control packages to control the same base tables:**
 - Enable application driven security policies
 - Development groups do not have to collude.
- **Group and manage packages with Policy Groups:**
 - Associate FGAC with specific applications
 - Default FGAC which always apply

ORACLE

1-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Fine-Grained Access Control Enhancements

When more than one Fine Grained Access Control (FGAC) has been defined on a table, a row must satisfy all requirements before being visible.

The FGACs are effectively “AND”-ed. This is not desirable if the table is to be used by users from separate groups, each group with a different FGAC. Prior to Oracle9i, one complicated FGAC that included all user groups’ requirements was defined.

Therefore, when multiple applications are sharing the same data, the idea of partitioned FGAC is introduced to allow application driven security policies.

Policy Groups

- **Policy groups are used to distinguish policies between different applications.**
- **The administrator designates an application context called a *driving context*, to indicate the policy group in effect.**
- **When the tables and views are accessed, the fine grained access control engine looks up the driving context to determine the policy group in effect.**
- **The feature will enforce all the associated policies that belong to the policy group.**

ORACLE

1-14

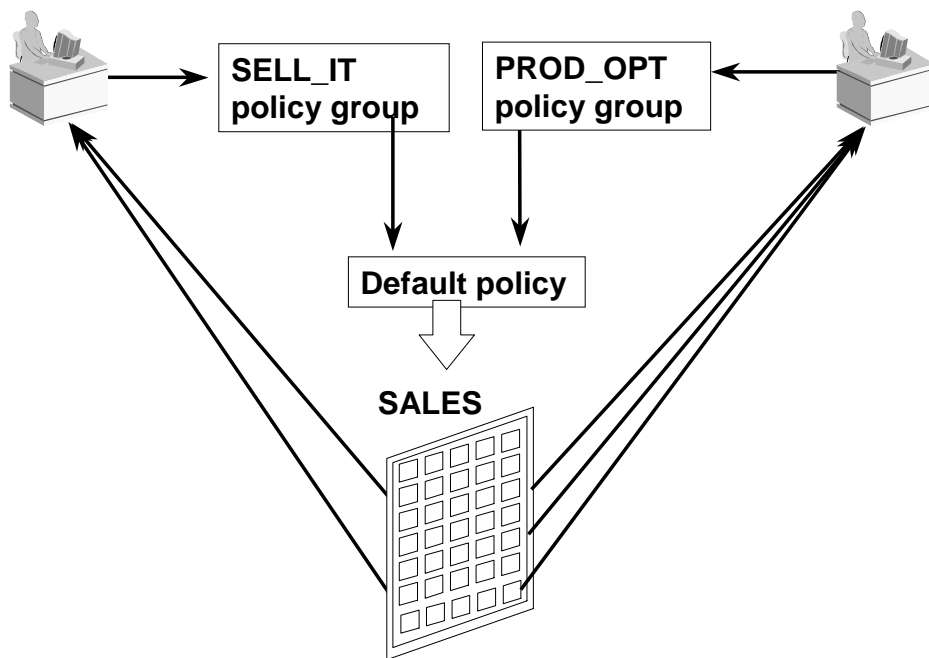
Copyright © Oracle Corporation, 2001. All rights reserved.

Policy Groups

With the partitioning of fine-grained access control policies, administrators can specify which policy group the policy falls into when adding policy to a table or view using the `ADD_POLICY_TO_GROUP` interface.

The policy group named `SYS_DEFAULT` is predefined. Policies defined in this policy group for a particular table or view are always executed along with the policy group specified by the driving context. For backward compatibility purpose, all Oracle8i fine grained access control policies are migrated into this default policy group to maintain the global policies before enabling the new feature.

Partitioned Fine-Grained Access Control



ORACLE

1-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Partitioned FGAC

For example, in case of a hosting environment, this feature is used to host the SALES table for two companies. Each bought separate applications, SELL_IT and PROD_OPT, wanting to enforce different security policies. The first company's SELL_IT application authorizes users to select only rows in the same region as themselves; that is, `SALES.CUST_ID + CUSTOMER.COUNTRY` must match the identifier associated to the user. The second company's PROD_OPT application limits row access to recent transactions, that is related by `SALES.TIME_ID`. In Oracle8i, integrating these two policies into one FGAC on the SALES table would require joint development, not really a feasible option if two products come from rival companies. By defining an application context to drive the enforcement of a particular set of policies to the base objects, each application can now implement a private set of security policies.

Fine-Grained Audit

- A tool to provide extensible intrusion detection, capturing the SQL statement, not the retrieved data
- Attach audit policy to table or view with **WHERE** condition on **SELECT** statements
- Oracle executes a user-defined audit event handler using autonomous transactions to process the event.
- An **AUDIT COLUMN** feature reduces false audit conditions.

ORACLE

1-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Fine-Grained Audit

Auditing in the DBMS is often used to monitor data access. Audit records are essential in proving the violation of access rights. The new Fine-Grained Audit (FGA) mechanism addresses the issues of more granular level of auditing.

The new auditing policy is based on simple user-defined SQL predicates on table objects as conditions for selective auditing. The predicate can specify auditing when a specified value is retrieved.

In addition to value based, there are also cases where the administrators are only interested in whether a certain column is being referenced or accessed. Because auditing is supported whenever a column is selected in any part of a DML statement, Oracle will audit the query.

In Oracle8i, audit options could only be set to monitor access of privileges or objects, and only a fixed set of facts, such as, userid, timestamp, and object name were recorded in the audit trail.

Fine-grained auditing can invoke a procedure as part of the audit.

Fine-Grained Auditing Implementation

- The security administrator uses `DBMS_FGA` to create an audit policy on the tables in question.
- List of defined policies in `DBA_AUDIT_POLICIES`
- Audit records are placed in `DBA_FGA_AUDIT_TRAIL`.
- Administrators can define *audit event handlers* to process the event, such as sending an alert page to the administrator.

ORACLE

1-17

Copyright © Oracle Corporation, 2001. All rights reserved.

FGA Implementation

Use the `DBMS_FGA` PL/SQL package interface to apply policies to tables or views:

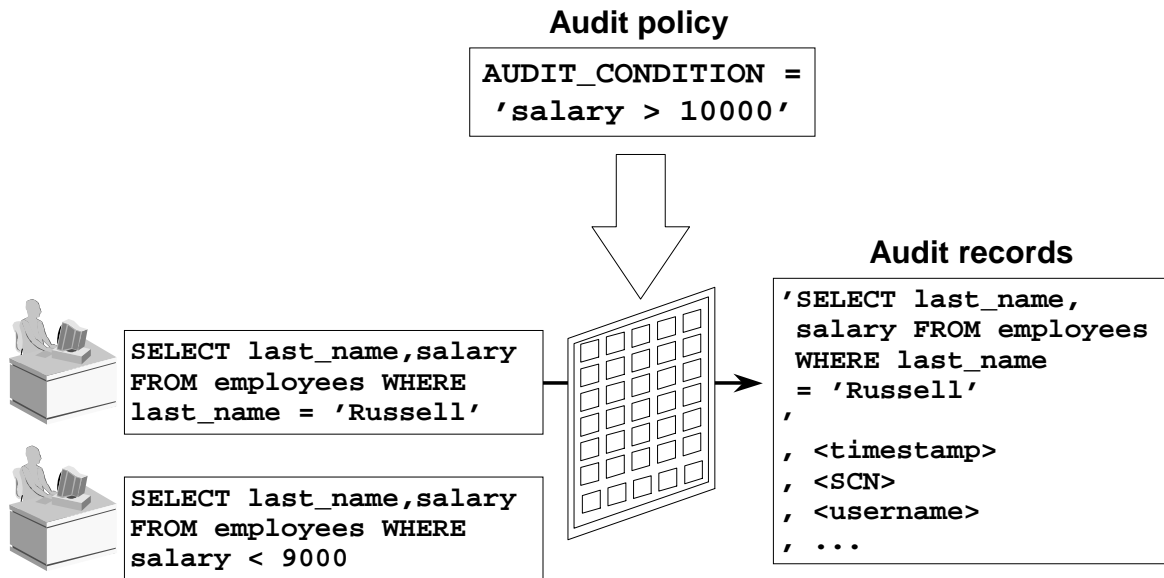
- `ADD_POLICY`
- `DROP_POLICY`
- `ENABLE_POLICY`
- `DISABLE_POLICY`

The Audit Record

`DBA_FGA_AUDIT_TRAIL` contains event entries consisting of username, SQL text, policy name, session id, timestamp, and other attributes.

The involved rows are referred to as *interested* rows.

Fine-Grained Auditing Example



Fine-Grained Auditing Example

Fine-grained auditing is available only on SELECT statements with a WHERE clause, and for one audit column only.

The triggering event above was not that the user accessed employee records for which salaries are greater than 10,000, but that the salary was actually returned to the user. A record is written to the audit trail that includes the complete SQL statement the user submitted, a timestamp, username, and other information.

Fine grained auditing does not automatically capture the values returned to the user. However, you may be able to use fine-grained auditing in conjunction with Flashback Query to re-create the records returned to the user.

Fine-grained auditing can function as an intrusion detection system for the database. For example, a developer could add an event handle to a policy to page an administrator if certain events occur. Non-SQL access is not audited. For example, if you use direct path export, which bypasses the SQL layer of the database, the audit condition is not triggered.

FGA Event Handler

Create an audit event handler:

```
CREATE PROCEDURE catchlog (schema_name VARCHAR2,  
    table_name VARCHAR2, policy VARCHAR2) AS  
BEGIN  
    -- send an alert note to my pager  
    UTIL_ALERT_PAGER('CatchLog:' || Table_name || SYSDATE);  
END;  
/
```

Add event handler policy:

```
DBMS_FGA.ADD_POLICY ( ...  
    HANDLER_SCHEMA=>'HRMGR', HANDLER_MODULE=>'CATCHLOG' );
```

ORACLE

Fine Grained Auditing Event Handler

In the example above, the user HRMGR creates a procedure CATCHLOG. The type of the arguments must be as shown, but the names can be edited.

The audit event handler is added into a audit policy by a privileged user.

When the first interested row is fetched, the audit event record is recorded and stored and the audit function HRMGR.CATCHLOG is called.

Encryption Enhancements

**DBMS_OBFUSCATION_TOOLKIT enhanced with
GETKEY to generate a key:**

- **More random than DBMS_RANDOM**
- **Federal Information Processing Standard - 140**

ORACLE

1-20

Copyright © Oracle Corporation, 2001. All rights reserved.

DBMS_OBFUSCATION_TOOLKIT Enhancements

The DBMS_OBFUSCATION_TOOLKIT packages require an encryption key as an attribute. If keys are weak (that is, easy to guess or predictable), it makes encryption much easier to break through cryptanalysis. Time of day, serial number of machines and the like are all *predictable* and thus cannot be used as random numbers for cryptographic keys.

DBMS_RANDOM in particular *cannot* be used as a secure random number generator since a given seed (input) will always produce the same output. In Oracle 9i, a Federal Information Processing Standard (FIPS) -140 certified random number generator (GETKEY) for secure key generation Secure random number generation makes implementation much easier

While key management is still programmatic, having a secure random number generator means that the solution is much more likely to be secure against cryptanalysis as long as keys are stored securely. That said, it would be difficult to recover data using a brute force attack if the keys are securely stored and randomly generated.

Note: Encrypting the data does not automatically make it secure; the security is only as good as security of the keys.

Oracle Label Security

- **Uses the Virtual Private Database features**
- **Enables the use of *labels* to ease identification of which rows can be retrieved**
- **Access rules are defined in *policies*.**

ORACLE

1-21

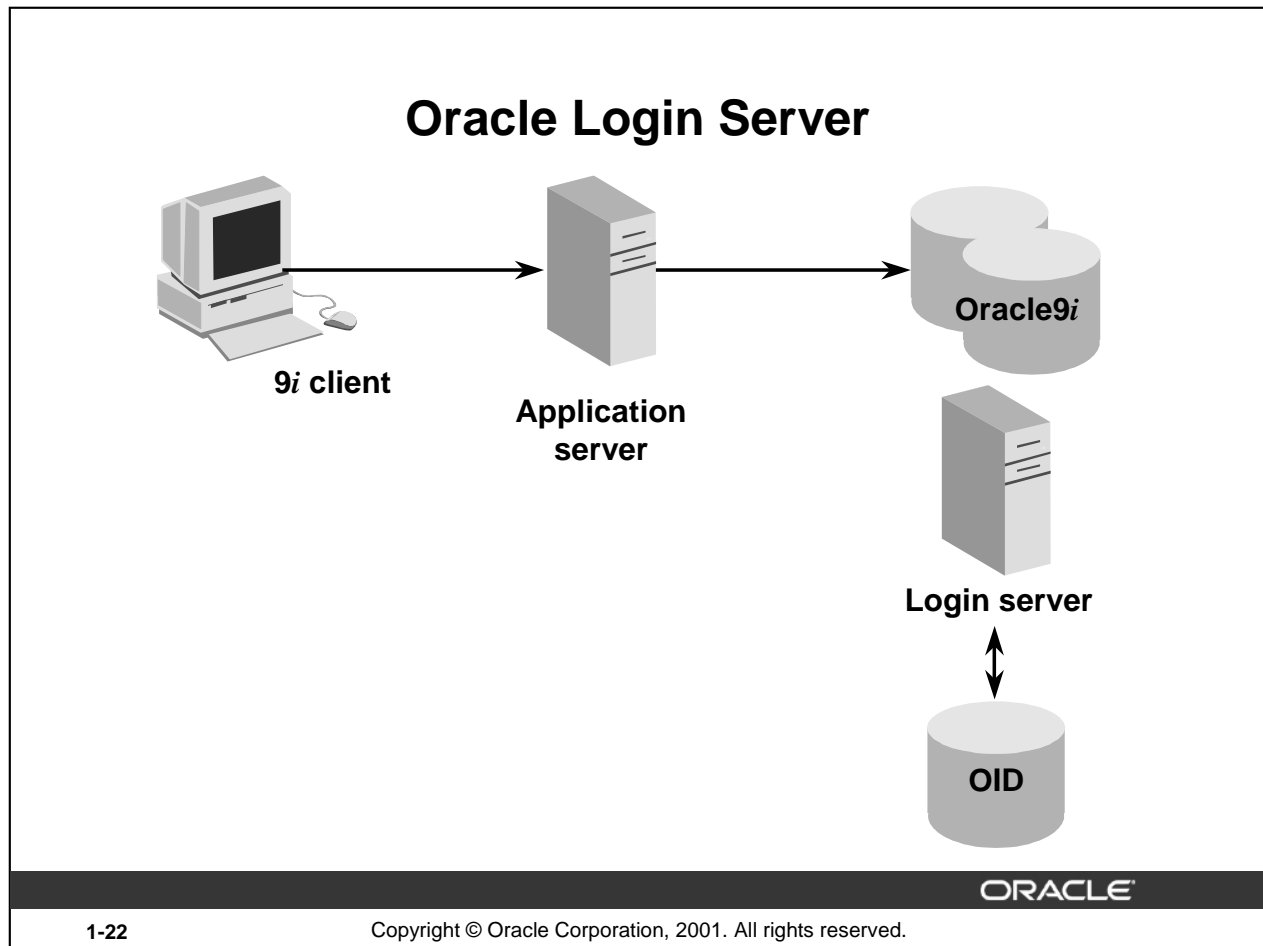
Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Label Security

Oracle Label Security enables application developers to add label-based access control to their Oracle9i applications. It mediates access to rows in database tables based on a label contained in the row, and the label and privileges associated with each user session. Oracle Label Security is built on the virtual private database technology of Oracle9i Enterprise Edition. It includes the Oracle Policy Manager, a graphical user interface for ease of administration.

Oracle Label Security is an optional product. Labels are created and managed through PL/SQL packages, which themselves are also protected through labels.

Oracle Label Security replaces and is similar to Trusted Oracle in earlier versions.



Oracle Login Server

The Login Server is included with Oracle Portal 3.0, which is part of Oracle9i Application Server.

The Login Server provides web-based single sign-on and integration with legacy applications. With single sign-on, users need maintain only a single strong user name/password account for accessing all Web applications throughout the enterprise. Applications integrated with Single Sign-On server securely delegate the user authentication process. Among other things, this enforces password rules, account lockouts based on repeated login failures, and auditing. The Single Sign-On server also has the ability to authenticate a user by means of an external repository such as LDAP or a database user repository, or by local authentication.

The Login Server is basically configured as an External Procedure, for which the for the platform appropriate .so or .dll file is supplied. A sql-script will create the required packages.

Oracle Enterprise Login Assistant

- **Can decrypt the wallet and establish an SSL connection to all PKI-enabled applications without requiring additional passwords**
- **Can update the directory password (OID only) and other related passwords stored in the directory**
- **Can upload an Oracle Wallet to the directory**

ORACLE

1-23

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Enterprise Login Assistant

Oracle Enterprise Login Assistant (ELA) is a client-side tool used to authenticate users to the enterprise. It can download an Oracle wallet from the directory. It can also decrypt the wallet and let users establish a seamless Secure Socket Layer (SSL) connection to all Public Key Infrastructure (PKI) enabled applications and databases within the enterprise, without requiring additional passwords. Enterprise Login Assistant can update the directory password (OID only) and other related passwords stored in the directory, and it can also upload an Oracle Wallet to the directory.

Enterprise User Security Enhancements

- **Integrate enterprise user management with proxy authentication**
- **Manage both password-based and SSL-authenticated users in a directory**
- **Improve ease of use**
- **Reduce processing overhead**
- **Provide backward compatibility for non-SSL clients**

Changes affect only the server side so older (pre Oracle9i) clients are interoperable.

ORACLE

1-24

Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits of Enterprise User Security Enhancements

Since enterprise users now require only one username and password (stored in OID) to access multiple databases, usability is improved. With the elimination of bi-directional certificate matching, processing overhead is reduced and performance improves.

Security administrators can set up enterprise users to access multiple databases with the same username and password. In addition, there is no requirement to install SSL on the client side, and no requirement to setup and maintain client-side wallets, thereby simplifying enterprise user administration.

Since there is no requirement for SSL or Oracle wallets on the client, all prior Oracle clients can function as enterprise users provided that they are password authenticated users. In addition, Oracle Enterprise Login Assistant lets users change passwords and utilize single sign-on whether they use Release 9i clients, Release 8i clients, or Release 8.0 or earlier versions.

Summary

In this lesson, you should have learned how to:

- **Use alternatives for the discontinued DBA access methods**
- **Choose a security product or component that facilitates the level of security appropriate to your data**

ORACLE

1-25

Copyright © Oracle Corporation, 2001. All rights reserved.

Selective Data Encryption provides a PL/SQL package for encrypting data inside the database, as well as Java-based encryption.

Virtual Private Database (VPD) provides non-bypassable, fine-grained, server-enforced security that eliminates the risks of application-based security.

Oracle Label Security (a server option) provides out-of-the-box VPD, and provides an extra layer of data protection by mediating access based on labels attached to data rows. For example, Top Secret, Engineering only, and so on.

Fine-Grained Auditing provides an extensible auditing mechanism to identify inappropriate data access and raise an alert.

Proxy Authentication enables you to know the identity of web users, limit privileges of middle tier servers, and provides accountability through auditing. Proxy authentication is supported for users known to the database, and users known only to an application, providing maximum flexibility for secure solutions.

Login Server provides web-based end users a single login account to access all web-based enterprise applications.

Enterprise User Security enables centralized management of users and their authorizations in an LDAP-based directory.

Public Key Infrastructure (PKI) integration facilitates deployment of *digital identities* for users, web servers, and data servers, including creation of public/private key pairs and X.509 digital certificates.

Practice 1-1 Overview

This practice covers the following topic:

Starting the database without using Server Manager or the INTERNAL keyword

ORACLE

2

General High Availability Technology

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Explain new features designed to harden the database against unplanned downtime**
- **Describe minimal I/O recovery**
- **Describe fast-start time-based recovery limit**
- **Explain the new Oracle Flashback feature**
- **Understand Resumable Space Allocation**
- **Understand the new Export and Import features**

ORACLE

Reducing Unplanned Downtime Overview

- **Unplanned downtime is the most disruptive event for an e-business.**
- **It can lead to lost revenue, lost productivity, disgruntled customers, even legal issues.**
- **The Oracle9i database builds upon the existing set of features provided to minimize unplanned downtime:**
 - **Speeding recovery**
 - **Reducing the impact of any type of failure on end-users**

ORACLE

Minimal I/O Recovery

- To reduce unplanned downtime, recovery time must be brought down to a minimum.
- The amount of I/O operations performed have a direct impact on the recovery time.
- Two elements control crash recovery time:
 - The time required to read change information from the redo log files
 - The time required to read, modify, and write the data blocks affected by those changes
- Oracle9i introduces a two-pass instance or crash recovery to reduce recovery time.

ORACLE

2-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Minimal I/O Recovery

The two-pass recovery explained here does not apply to media recovery. This feature is only used during instance or crash recovery.

Minimal I/O Recovery

- **The log files often contain changes for data blocks which were not dirty at the time of failure.**
- **In Oracle9i, additional information is recorded in the logs to indicate which blocks have been written to disk successfully.**
- **These blocks are not processed during recovery.**
- **The overall recovery time is reduced.**
- **This feature is enabled by default and no configuration is required by the database administrator (DBA).**

ORACLE

Minimal I/O Recovery (continued)

To bound the time it takes to recover from a crash, two time elements must be controlled:

- The time required to read the log files, which is dependent on the data transfer rate of the log device, and the amount of log that is required to be read during the recovery process.
- The time required to read and write the data blocks which were modified in the buffer cache at the time of failure. This is done by limiting the number of modified blocks in the cache to the number that can be read and written during the time allocated for that phase of recovery.

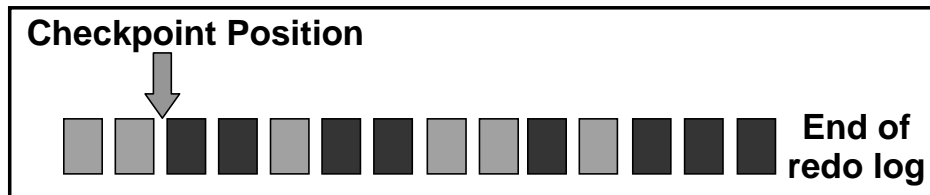
However, the redo log may contain entries recording changes made to data blocks which may not be dirty in the buffer cache at the time of failure. These data blocks, which were successfully written to disk before the failure, no longer need to be read and inspected during recovery, possibly saving a significant amount of time.

This new feature reads the logs twice. The first time determines which blocks need recovery and the second time applies only the minimum required changes. The first sequential read is very fast and the savings in workload reduction easily offset the cost of this additional read. The result is improved overall recovery time.

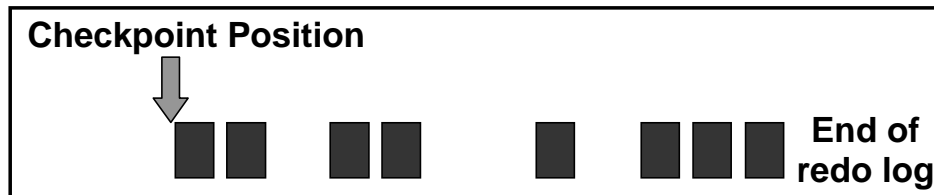
Two-pass recovery reduces recovery time whenever the I/O time for doing the additional data block reads is greater than that of doing the second pass on the log. As the data block reads are essentially random, while the log reads are sequential, this will normally be the case. The degree of overall improvement will therefore vary from one instance to another.

Minimal I/O Recovery

1. Determines which records are needed to recover in-doubt blocks



2. Applies only the records needed for recovery



ORACLE

Minimal I/O Recovery (continued)

On this diagram, the green squares (light gray) represent change vectors in the redo log that are not required for recovery because they correspond to blocks that were already written to disk at the time of failure.

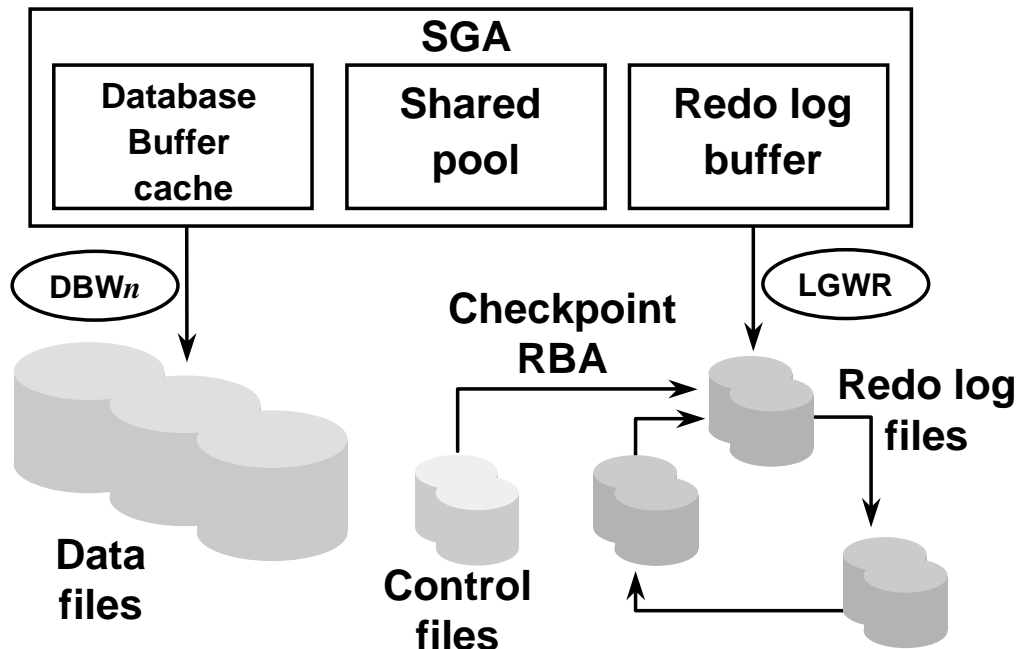
The red squares (dark gray) represent change vectors associated with doubtful blocks which may need recovery. This is the subset of change vectors that will be applied.

The performance impact caused by the additional information recorded in the logs is negligible.

The sequential read time consumed reading the logs for the first time is estimated to be small in relation to the savings in read time by not reading database blocks which are not needed for instance or crash recovery.

The first sequential read does not touch the data blocks at all, it only reads the online log files. The information from the first pass is kept in the PGA and is used to process the data blocks in the second pass.

Fast-Start Time-Based Recovery Limit



ORACLE

2-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Fast-Start Time-Based Recovery Limit

During recovery, the Oracle9i instance replays changes beginning at the checkpoint redo byte address (Checkpoint RBA). The checkpoint RBA is stored in the control file. When recovery is required, the checkpoint RBA determines the location in the redo stream from which to start applying recovery.

Advancing the position of the checkpoint RBA reduces recovery time. In order to advance the checkpoint RBA, dirty blocks in the buffer cache have to be written to the data files. This operation, known as a checkpoint, can have an adverse effect in performance if done with excessive frequency.

As it often occurs, a balance must be found between performance and recoverability.

Fast-Start Time-Based Recovery Limit Overview

- Many Service Level Agreements include a bound on the Mean Time To Recover (MTTR).
- The DBA must be able to reliably set a limit on the time it will take to recover the database.
- Oracle9i introduces Fast-Start Time-Based Recovery.
- This feature lets a DBA specify a target for recovery time in seconds.
- The Oracle9i instance automatically determines appropriate values for internal settings to meet the specified target.

ORACLE

2-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Fast-Start Time-Based Recovery Limit - Overview

It is more user-friendly to set an MTTR limit in seconds. Previously, trying to set the MTTR was more *difficult* and less precise.

Fast-Start Time-Based Recovery Limit

- **Recovery time is dependent on:**
 - Time to read the log files
 - Time to process the recovered data blocks
- **LOG_CHECKPOINT_INTERVAL** bounds the number of redo records to be read.
- **FAST_START_IO_TARGET** controls the number of data blocks to be recovered.
- **However, a DBA cannot set a MTTR limit in seconds using these parameters alone.**

ORACLE

2-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Fast-Start Time-Based Recovery Limit

Note: LOG_CHECKPOINT_TIMEOUT limits the number of seconds between the most recent redo record and the checkpoint. Nevertheless, it is not precise enough to set a limit on the MTTR target.

Fast-Start Time-Based Recovery Limit

- The new parameter **FAST_START_MTTR_TARGET** allows a DBA to specify the estimated number of seconds crash recovery should take.
- Internally this is translated to a setting for:
 - **FAST_START_IO_TARGET**
 - **LOG_CHECKPOINT_INTERVAL**
- This feature greatly simplifies and increases the accuracy of bounding database recovery time.
- **FAST_START_MTTR_TARGET** is a dynamic parameter.

```
ALTER SYSTEM SET FAST_START_MTTR_TARGET = 60;
```

ORACLE

2-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Fast-Start Time-Based Recovery Limit (continued)

The value of the **FAST_START_MTTR_TARGET** parameter is the estimated mean time to recover (MTTR) of crash recovery in a single instance configuration. With Real Application Clusters, the worst case expected MTTR for crash recovery is the sum of all **FAST_START_MTTR_TARGET** parameters set in all open instances (read/write open). The actual MTTR may be less because *initialization* and *file open* occur only once.

However, the expectation is that in a Real Application Clusters environment, the MTTR will typically be less, because the failure of a single instance, recovered by an already-open instance will be the normal recovery situation. Crash recovery covers the failure of all nodes in the Real Application Clusters environment, an extremely low probability occurrence.

The parameter **FAST_START_MTTR_TARGET** can take an integer in the range 0 to 3600 seconds. A value of 0, which is the default, indicates that the control of the recovery time is not done with this parameter. The recommended value depends of the size of the SGA, the load on the database, and many other varying factors, including the service level agreement (SLA) for that site. The idea is to find the value that brings recovery time in line with the SLA without incurring an excessive performance penalty.

The performance degradation is detected by looking at the number of additional blocks checkpointed as a consequence of a very restrictive MTTR target. Checkpoint statistics are provided in the Statspack output (physical checkpoint writes versus physical writes) as well as in the **V\$INSTANCE_RECOVERY** view (**ckpt_block_writes**).

Fast-Start Time-Based Recovery Limit

- During recovery, the Oracle9i instance replays changes beginning at the checkpoint redo byte address (Checkpoint RBA).
- Advancing the checkpoint position controls recovery time.
- Until now, four parameters controlled checkpoint advance:
 - DB_BLOCK_MAX_DIRTY_TARGET
 - FAST_START_IO_TARGET
 - LOG_CHECKPOINT_INTERVAL
 - LOG_CHECKPOINT_TIMEOUT

ORACLE

2-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Fast-Start Time-Based Recovery Limit (continued)

DB_BLOCK_MAX_DIRTY_TARGET: specifies the maximum number of dirty buffers that are allowed in the buffer cache.

FAST_START_IO_TARGET: specifies the number of data blocks to be recovered. Prior to Oracle9i this parameter controlled the number of data blocks to be read and written. With two-pass recovery, this is equal to the number of blocks to be recovered.

LOG_CHECKPOINT_INTERVAL: specifies the maximum number of redo records between the checkpoint position and the end of the log.

LOG_CHECKPOINT_TIMEOUT: specifies in seconds, how long ago the redo record at the checkpoint position was written.

Note: While not a parameter, the checkpoint advance is also controlled by 90 percent of the smallest log size. Oracle enforces this behavior by ensuring the number of redo blocks between the checkpoint and the most recent redo record is less than 90 percent of the size of the smallest log. If the value of LOG_CHECKPOINT_INTERVAL is less than 90 percent of the size of the smallest log, then the size of the smallest log file does not influence checkpointing behavior.

Changes to Previous Parameters

- **DB_BLOCK_MAX_DIRTY_TARGET has been removed**
- **If FAST_START_IO_TARGET or LOG_CHECKPOINT_INTERVAL are specified, they override the values calculated by FAST_START_MTTR_TARGET.**
- **There is no change to the behavior of LOG_CHECKPOINT_TIMEOUT.**

ORACLE

2-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Changes to Previous Parameters

The new parameter FAST_START_MTTR_TARGET is internally translated into two settings: one for FAST_START_IO_TARGET and another one for LOG_CHECKPOINT_INTERVAL. If these last two parameters are not explicitly set, the calculated values take effect.

If explicit values are set for FAST_START_IO_TARGET or LOG_CHECKPOINT_INTERVAL however, the internally calculated value is ignored for the corresponding parameter and the explicit setting takes place.

The algorithm takes into account such tasks as initialization, file open, reading the log, reading the data blocks from the data files, and writing the data blocks back to the data files. It is an adaptive algorithm, and initially uses internal estimates for these operations, but substitutes real world data as it becomes available through system monitoring.

Therefore the estimates become more accurate over time, as the server learns more about its environment and expected I/O times. Because manually measuring the time it takes to complete these operations, and calculating values for parameters controlling recovery time is a complex task, this feature greatly simplifies and increases the accuracy of bounding instance recovery time.

Changes to V\$INSTANCE_RECOVERY

- **Three new columns have been added.**
- **These columns supercede the previous information which is only retained for compatibility purposes.**
- **The new columns are:**
 - **TARGET_MTTR: user setting of parameter FAST_START_MTTR_TARGET**
 - **ESTIMATED_MTTR: current estimated MTTR based on the number of dirty buffers and number of log blocks**
 - **CKPT_BLOCK_WRITES: number of blocks written by checkpoint writes**

ORACLE

2-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Changes to V\$INSTANCE_RECOVERY

The Oracle9i instance initially uses internal estimates for the I/O rates during recovery in order to calculate values for the parameters. Through system monitoring, it measures actual I/O rates and uses this information to make more accurate calculations for subsequent recoveries. Thus, the MTTR estimate becomes more accurate over time.

The Oracle9i instance adjusts the checkpointing rate to meet the target specified by the parameters. However, this is not a hard target. Therefore, it is quite possible that the estimated MTTR is not equal the target set.

The V\$INSTANCE_RECOVERY view is used to monitor checkpointing, and its impact on recovery time. Every 30 seconds, the Oracle9i Database calculates an estimate of the current MTTR and places this value in V\$INSTANCE_RECOVERY. This allows the DBA to monitor the current estimated MTTR, and compare it to the target specified by FAST_START_MTTR_TARGET.

Setting a very small value for MTTR can have an adverse effect on performance because unnecessary checkpoints will be generated. To help the administrator monitor the impact of small values of this parameter, the view also displays the additional database writes due to checkpointing in the column CKPT_BLOCK_WRITES.

Oracle Flashback Overview

- **This feature allows users to see a consistent view of the database at a point in the past.**
- **Users can specify this read-only view based on a system time or a system change number (SCN).**
- **Only transactions committed up until that time are visible.**
- **Possible applications are:**
 - **Self-service repair**
 - **Packaged applications like email**
 - **Decision support systems for trend analysis**

ORACLE

2-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Flashback Overview

The database system guarantees to the user that all queries submitted to the database see the consistent version of the database as of a user specified time (this is a server-side system time). All transactions committed until then are visible. Those committed after that are not.

Applications can be executed on the version of the database as of a certain time. For example, a windows application for customer information may have a button that allows the user to move back to an earlier point in time to show the account balance as of a certain time. Previously, much of this information had to be saved away by the application. This feature allows accessing information in the past, even when it was not explicitly saved away.

There may be several reasons why users would like to query past data. One important application would be self-service repair, when some important rows were accidentally deleted from a table and the users wanted to recover them. To do the repair, they could possibly move backwards in time and see the missing rows and re-insert the deleted row into the current table. Care must be taken not to introduce logical inconsistencies.

Other potential benefits include packaged applications such as email and voice mail. The users may have accidentally deleted a mail or a voice message by pressing the wrong key. Using temporal access, they would be able to restore the deleted mail or message by moving back in time and re-inserting the deleted message into the current message box.

Oracle Flashback

- **Oracle Flashback leverages the Automatic Undo Management functionality.**
- **Undo information is kept for a specified retention interval at system level.**
- **Queries addressed to a time within the retention interval have enough undo information to reconstruct the snapshot.**
- **Oracle Flashback is enabled at a session level.**
- **PL/SQL cursors opened prior to disabling Flashback may be used to perform DML.**
- **SMON keeps a table to map system time to SCNs:
SMON_SCN_TIME**

ORACLE

2-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Flashback Overview (continued)

Other beneficiaries include Decision Support Systems (DSS)/Online Analytical Processing (OLAP) applications that have to perform long running transactions. They may want to perform data analysis and modeling on data as of some time in the past, for example, to track seasonal demand on sunscreens.

The package `DBMS_FLASHBACK` is provided to enable Flashback at the specified system time or SCN. Once Flashback has been enabled in a session, the user can execute arbitrary queries or PL/SQL packages. All the queries against the database run as of the specified time/SCN. Thus, the user session sees the version of the database as of the specified time/SCN. When a session ends by disconnect or another connect, Flashback is automatically turned off. The user can explicitly turn off Flashback by using the `disable` method in the package. The package may be used from inside logon triggers to automatically enable Flashback without changing application code.

PL/SQL cursors that are opened in Flashback mode return rows as of the time/SCN being used by Flashback at the time of opening the cursors. Different concurrent sessions, or connections, in the database may perform Flashback to different time/SCNs. DML and DDL operations and distributed operations are not allowed while a session is running in Flashback mode. PL/SQL cursors opened prior to disabling Flashback can be used to perform DML.

Oracle Flashback Overview (continued)

This feature leverages the Automatic Undo Management functionality, described later in this course, which retains the undo segment for a user specified retention interval.

Therefore, all queries addressed to a system time which is not earlier than a retention interval from the current time have enough undo information to reconstruct the snapshot. The amount of disk space required will depend on how far back you intend to go.

When enabling Flashback using system time, the database chooses an SCN that was generated at a time that was within five minutes prior to the specified time. For finer grain control of Flashback the user can enable using an SCN.

Oracle Flashback

- The DBA must set the undo retention interval long enough to be able to reconstruct the data.

```
ALTER SYSTEM SET UNDO_RETENTION = <seconds>;
```

- The package DBMS_FLASHBACK provides the required interface.

```
call dbms_flashback.enable_at_time('9-NOV-01:11:00:00');  
SELECT * FROM employees;  
call dbms_flashback.disable();
```

- Users need execute privilege on DBMS_FLASHBACK to use the feature.

ORACLE

2-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Flashback Overview (continued)

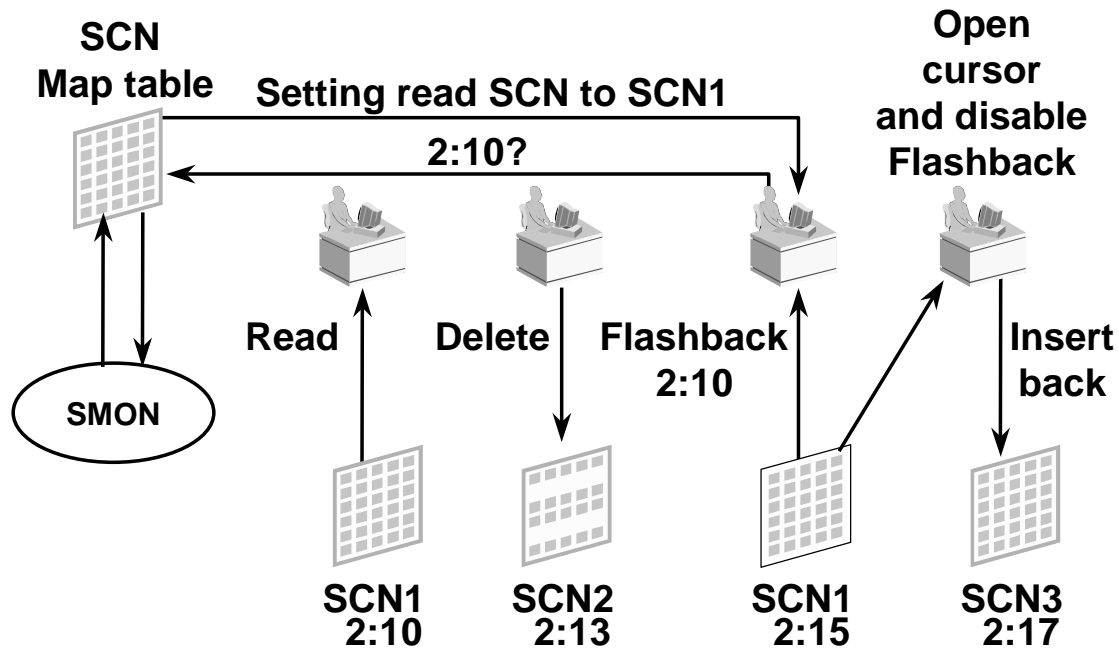
The flashback view is bounded by the undo retention interval. This is specified in seconds by the database administrator and the maximum value is unlimited.

The package DBMS_FLASHBACK provides the following interface:

- **ENABLE_AT_TIME**: Enables Flashback for the entire session. The snapshot time is set to the SCN that most closely matches the specified timestamp.
- **ENABLE_AT_SYSTEM_CHANGE_NUMBER**: Takes a system change number (SCN) as an Oracle number and sets the session snapshot to the specified number.
- **GET_SYSTEM_CHANGE_NUMBER**: Returns the current SCN as an Oracle number. With this interface, users can obtain the current change number and store it away for later use.
- **DISABLE**: This procedure disables the Flashback mode for the entire session and brings you back to the state of data at the current time.

Note: The DBMS_FLASHBACK.ENABLE procedure cannot be used while a transaction is active. Also, the DBMS_FLASHBACK package cannot be executed as user SYS.

Oracle Flashback



ORACLE

2-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Flashback Overview (continued)

The slide illustrates one usage of Flashback:

- At time 2:10 the user can read certain data from the table.
- At time 2:13 some rows are accidentally deleted.
- At time 2:15 the user enables Oracle Flashback and creates a view of the data as it looked at time 2:10. The SMON process keeps a conversion table and resolves the specified time to the corresponding SCN. A PL/SQL cursor can then be created joining this read-consistent view at that SCN with the current table itself so that the cursor contains only the missing rows.
- At time 2:17, with the cursor already created, the user can disable Oracle Flashback to return to the current point in time. Then the user fetches the rows from the cursor and inserts them back into the table.

Note: All the DML occurs in the present state of the data. The Flashback functionality does not work to reverse DDL statements such as a drop or truncate command, for example. This is because the rollback segments do not store enough information to recreate the previous state of the object. Also, the Flashback functionality does not apply to PL/SQL packages, procedures, or functions. Invocations to any program object that does not enable Flashback will always use the current version of these objects even when Flashback is enabled outside the program object call.

Oracle Flashback and Oracle LogMiner

- Both tools can be used interchangeably to undo certain user errors.
- Flashback works with the current undo.
- Its scope depends on the retention interval, which is typically short because of space requirements.
- LogMiner works with current and old redo logs and, therefore, can reach much further back.
- Flashback is faster and easier to set-up.
- LogMiner can be used for other purposes and provides a GUI interface, but it does not support all data types.

ORACLE

Resumable Space Allocation Overview

Resumable Space Allocation provides:

- **The ability to suspend and resume execution of large database operations in the event of repairable failure.**
- **Support for errors related to space limits and out-of-space conditions.**
- **An opportunity for the DBA to take the corrective steps to resolve the error condition.**
- **Suspended statements that automatically continue operation.**

ORACLE

2-20

Copyright © Oracle Corporation, 2001. All rights reserved.

Resumable Space Allocation Overview

Resumable Space Allocation provides the ability to suspend and resume execution of large database operations in the event of repairable failure. Resumable Space Allocation feature supports a class of errors related to space limits and out-of-space conditions.

Large operations routinely run out of space or reach space limits after executing for a long time. When this condition occurs, the operation is rolled back and the error is returned to the user. The rollback takes time comparable to how long the operation ran until it failed.

This feature suspends large operations that run into these problems. Suspending the operation gives the DBA an opportunity to take the corrective steps to resolve the error condition. Once the error condition disappears, the suspended statement automatically resumes the statement's execution.

Prior to this feature, customers used their home grown mechanisms to deal with failure. They divided the operation into smaller chunks and wrote records that track the progress of the operation. This feature enables customers to write applications without worrying about running into common space related errors. These errors may be handled and fixed while the large operation is in progress.

It is desirable for the user to register a procedure to be automatically executed when the statement gets suspended. This gives the user the opportunity to detect the problem and take appropriate actions.

Life Cycle for Resumable Space Allocation

- You enable Resumable Space Allocations using the **ALTER SESSION** command.
- A statement is suspended when one of the following conditions occurs:
 - Out of space condition
 - Maximum number of extents reached condition
 - Space quota exceeded condition
- When a statement is suspended:
 - The error is reported in the alert log
 - A system event trigger, after suspend, can be executed
- When the error condition disappears, the suspended statement automatically resumes.

ORACLE

2-21

Copyright © Oracle Corporation, 2001. All rights reserved.

Life Cycle for Resumable Space Allocation

Following are the characteristics of Resumable Space Allocation execution:

- A statement executes in a Resumable Space Allocation mode only when the client explicitly enables Resumable Space Allocation semantics for the session using the **ALTER SESSION** command
- In this mode, a statement is suspended when one of the following conditions occur (these conditions do not signal an error message):
 - Out of space condition: The operation cannot acquire any more space from a tablespace.
 - Max extents reached condition: The number of extents in a table, index, temp segment, rollback segment, cluster, LOB, table partition, index partition equals the maximum extents defined on the object.
 - Space quota exceeded condition: The user has used up his quota.
- On suspending a statement execution, there are mechanisms to perform user supplied operations, log errors, and query the status of the statement execution. When a statement is suspended the following actions are taken:
 - The error is reported in the alert log

Life Cycle for Resumable Space Allocations (continued)

- If the user registered a trigger on the system event, after `suspend`, the user trigger is executed. The user supplied PL/SQL procedure may access the error message data using the `DBMS_RESUMABLE` packages and `USER (DBA) _RESUMABLE` view.
- PL/SQL stored procedures, Java stored procedures, DMLs, DDLs, and queries running out of sort space are all able to execute in Resumable Space Allocation mode.
- DDLs of interest are:
 - `CREATE TABLE AS SELECT`, `CREATE INDEX`, `ALTER INDEX REBUILD`
 - `ALTER TABLE {MOVE | SPLIT} PARTITION`
 - `ALTER INDEX {REBUILD | SPLIT} PARTITION`
 - `CREATE MATERIALIZED VIEW [LOG]`
- Suspending a statement automatically results in suspending the transaction. Thus all transactional resources are held through a statement suspend and resume.
- When the error condition disappears whether as a result of user intervention or not (For example, sort space released by other queries), the suspended statement automatically resumes execution. A statement may be suspended and resumed multiple times during its execution.
- A suspension time out interval is associated with Resumable Space Allocation mode. A statement that is suspended for the timeout interval wakes up and returns the exception to the user.
- A suspended statement may be forced to throw the exception using `DBMS_RESUMABLE.ABORT ()` procedure which might be called by either DBA or the user who issued the statement.

Note: If a statement executes in Resumable Space Allocation mode, and performs an operation on a remote database, the remote operation is not executed in Resumable Space Allocation mode.

Resumable Space Allocation Operations

- **Queries:** Select statements that run out of temporary space.
- **Data Manipulation Language commands:** Insert, Update, Delete may be Resumable Space Allocation operations.
- **Import/Export** when invoked with the Resumable Space Allocation option.
- **SQL*Loader** when invoked with the Resumable Space Allocation option.
- **Various Data Definition Language commands.**

ORACLE

2-23

Copyright © Oracle Corporation, 2001. All rights reserved.

Resumable Space Allocation Operations

Broadly, Selects, DMLs, and certain DDLs, PL/SQL, and Java stored procedures, functions, and anonymous server side PL/SQL blocks executing in Resumable Space Allocation mode can be suspended whenever the underlying SQL statement is suspended. SQL statements include OCI calls like `OCIStmtFetch()`, `OCIStmtExecute()`, and so on. Following are Resumable Space Allocation SQL statements:

- **Queries:** Select statements that run out of temporary space (for sort areas) are candidates for Resumable Space Allocation execution. When using OCI, the calls, `OCIStmtExecute()`, `OCIStmtFetch()`.
- **DMLs:** Insert, Update, Delete may be Resumable Space Allocation operations. The interface used to execute them does not matter.
- **Import/Export.** Resumable Space Allocation mode will be an option to both Import and Export, like SQL*Loader. The new parameters are: `RESUMABLE=Y` to wait for a tablespace full event, and `RESUMABLE_TIMEOUT` to specify a timeout.
- **DDLs:** `CREATE TABLE AS SELECT`, `CREATE INDEX`, `INDEX REBUILD`, `ALTER TABLE {MOVE | SPLIT} PARTITION`, `ALTER INDEX {REBUILD | SPLIT} PARTITION`, and `CREATE MATERIALIZED VIEW [LOG]`.

Enable Session Resumable Space Allocation

```
ALTER SESSION ENABLE RESUMABLE  
TIMEOUT 60 NAME 'Starting Point';
```

```
ALTER SESSION DISABLE RESUMABLE;
```

```
ALTER SESSION ENABLE RESUMABLE NAME 'new name';
```

```
call DBMS_RESUMABLE.SET_TIMEOUT(...);
```

```
call DBMS_RESUMABLE.SET_SESSION_TIMEOUT(...);
```

ORACLE

2-24

Copyright © Oracle Corporation, 2001. All rights reserved.

Enable Session Resumable Space Allocation

A new `ALTER SESSION ENABLE RESUMABLE [TIMEOUT timeout] [NAME name]` statement allows you to enable SQL statements to be resumable when they are invoked within this session. `ALTER SESSION DISABLE RESUMABLE` will disable resumability of subsequent SQL statements invoked in the same session.

- **timeout:** A suspended statement is aborted automatically if the error is not fixed within the specified number of seconds.
- **name:** A user defined text string which is inserted into `USER[DBA]_RESUMABLE` view to help you identify a specific statement. The length may not exceed 4000 bytes (maximum of `VARCHAR2`.)

The default for a new session is `DISABLE`.

Examples:

```
SQL> ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600  
2 NAME 'CRM application';
```

```
SQL> ALTER SESSION DISABLE RESUMABLE;
```

Note: The default value for `NAME` is:

- User `USERNAME (USERID)`
- Session `SESSIONID`
- Instance `INSTANCEID`

Enable Session Resumable Space Allocation (continued)

Users can also change resumable name within a session. Example:

```
SQL> ALTER SESSION ENABLE RESUMABLE
  2  NAME 'delete from table';
...
SQL> ALTER SESSION ENABLE RESUMABLE
  2  NAME 'insert into table';
```

In order to reduce the use of initialization parameters, Oracle hard coded the default timeout to 7200 seconds (2 hours.) This timeout can be changed by privileged users by using one of the following methods:

```
SQL> ALTER SESSION ENABLE RESUMABLE TIMEOUT timeout;
SQL> EXECUTE DBMS_RESUMABLE.SET_SESSION_TIMEOUT -
  2  (sessionID, timeout);
SQL> EXECUTE DBMS_RESUMABLE.SET_TIMEOUT(timeout);
```

DBAs can change the default system timeout by creating a system wide AFTER SUSPEND trigger to set it. For example, to set system wide default timeout to one hour (concept introduced in the rest of this lesson):

```
SQL> CREATE OR REPLACE TRIGGER resumable_default_timeout
  2  AFTER SUSPEND ON DATABASE
  3  BEGIN
  4      DBMS_RESUMABLE.SET_TIMEOUT(3600);
  5  END;
  6  /
```

Note: Whenever a suspended statement times out, the user receives the following error message: ORA-30032: the statement has timed out. For compatibility reason, users can disable this behavior by setting the following event:

```
event="30030 trace name context forever, level 1"
```

DBMS_RESUMABLE Package

These are the DBMS_RESUMABLE procedures:

- **ABORT(sessionID)**
- **GET_SESSION_TIMEOUT(sessionID)**
- **SET_SESSION_TIMEOUT(sessionID, timeout)**
- **GET_TIMEOUT()**
- **SET_TIMEOUT(timeout)**
- **SPACE_ERROR_INFO(error_type, object_type, object_owner, table_space_name, object_name, sub_object_name)**

ORACLE

2-26

Copyright © Oracle Corporation, 2001. All rights reserved.

DBMS_RESUMABLE Package

The new DBMS_RESUMABLE package helps you to control Resumable Space Allocation mode:

- **ABORT(sessionID)**: Aborts a suspended statement. 'sessionID' is the session ID in that the statement is executed. For a parallel DML/DDL, 'sessionID' is any session ID which participates in the parallel DML/DDL. It is guaranteed that this operation will always succeed. It can be called either inside or outside of the AFTER SUSPEND trigger. The user then receives the following error message: ORA-30031: the statement has been aborted.
- **GET_SESSION_TIMEOUT(sessionID)**: This function returns current timeout value in seconds of statements for session with sessionID.
- **SET_SESSION_TIMEOUT(sessionID, timeout)**: This procedure sets the timeout of statements for session with sessionID. The parameter 'timeout' is in second. The new timeout setting applies to the session immediately.
- **GET_TIMEOUT()**: This function returns current timeout value of statements for the current session. The returned value is in second
- **SET_TIMEOUT(timeout)**: This procedure will set timeout of statements for current session. The parameter 'timeout' is in second. The new timeout setting will apply to the session immediately.

DBMS_RESUMABLE package (continued)

- `SPACE_ERROR_INFO`: All above parameters are OUT parameters. This function returns a Boolean value and is used to get information about the statements errors.

Note: It is not possible to use this package to enable Resumable Space Allocation mode in a particular session. Instead, DBAs have the possibility to create a logon trigger that will enable this feature by using an `ALTER SESSION` command.

AFTER SUSPEND System Event

- Automatically generated when a statement encounters a correctable error:

```
create or replace trigger res_default
after suspend on database
DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
begin
  /* send an email to notify DBA */
  COMMIT;
end;
```

- SQL statements executed within an AFTER SUSPEND trigger are always non-resumable

ORACLE

2-28

Copyright © Oracle Corporation, 2001. All rights reserved.

AFTER SUSPEND System Event and Trigger

When a statement is suspended, the (correctable) error is not raised to the client. An alternative method of notifying and detecting suspended statements is provided:

A new system event, *After Suspend* is available. Users can create a trigger on this system event. Whenever a statement executed by that user is suspended, the *After Suspend* trigger is executed. Inside the trigger, the user can access information regarding the suspended statement from the `dbms_resumable` package. The information includes user supplied session name, the Oracle error number and message, and so on. The trigger may execute a trusted callout. The trigger must be declared as an autonomous transaction. The autonomous transaction uses a rollback segment in the SYSTEM tablespace. This reduces the chance of running into the same error as the user statement. Also, the user is advised to perform operations in the trigger that are arranged to succeed. For example, the trigger operations can be run using a separate tablespace such as SYSTEM. The user may still run out of sort space, though. The trigger can use the `dbms_resumable` package to abort the statement, if appropriate.

Note: The trigger should be created under SYS schema.

AFTER SUSPEND System Event and Trigger (continued)

The trigger may encounter the following errors:

- Deadlock with a lock held by the user transaction (which is executing the statement). This can happen if the trigger attempts to update a row locked by the user transaction. The Oracle server detects the deadlock, rolls back work done in the trigger and throws the original exception to the user. Thus, if the trigger deadlocks (self locks) with the user transaction, the result is that of not suspending the statement. Also, the deadlock is pushed under the user error.
- Out of space or space limit reached condition. If the trigger throws these errors, it does not get suspended. Instead the trigger and the statement are aborted, as above.

Note: Users may have exception handlers inside the trigger to clear the error, if they do not want the suspended statement to be aborted.

RESUMABLE System Privilege

- **RESUMABLE system privilege allows you to execute statements in Resumable Space Allocation mode.**
- **To grant RESUMABLE privilege to a user, a DBA can issue:**

```
SQL> GRANT RESUMABLE TO hr;
```

- **To revoke this privilege, the DBA can issue:**

```
SQL> REVOKE RESUMABLE FROM hr;
```

ORACLE

RESUMABLE System Privilege

Because a suspended statement will hold up some system resources, a new RESUMABLE system privilege is added so that DBA can decide who can execute statements in Resumable Space Allocation mode.

If a user issues a statement without RESUMABLE privilege, the system will return an error if the statement generates a correctable error.

DBA_RESUMABLE Dictionary View

SESSION_ID	Session identifier of the statement
INSTANCE_ID	Instance number of the statement
SQL_TEXT	First 1000 characters of the statement
NAME	The name given to the statement
STATUS	RUNNING, SUSPENDED, COMPLETED, ABORTED, TIMEOUT
ERROR_NUMBER	Error code of the last correctable error (1)
ERROR_MSG	Error message corresponding to (1)
START_TIME	Start time of the statement
SUSPEND_TIME	Last time the statement was suspended
RESUME_TIME	Last time the statement was resumed

ORACLE

2-31

Copyright © Oracle Corporation, 2001. All rights reserved.

DBA (USER) _RESUMABLE Dictionary View

The DBA (USER) _RESUMABLE views display the set of Resumable Space Allocation statements in the system. This view is built on top of some SGA data structures (v\$resumable and v\$session_wait). The column STATUS indicates the status of the statement, such as RUNNING, SUSPENDED, and so on.

The view can be used by the AFTER SUSPEND trigger or by another session to monitor the progress of statements submitted to the database.

The view contains all currently executing or suspended statements in Resumable Space Allocation mode. Because information in this view is not persistent, it cannot be accessed across database shutdown or startup.

There are columns indicating the time started, time suspended, time completed or aborted for each statement.

Note: When a statement is suspended, the session that invokes that statement is put in waiting state and a new row with event 'Suspended on Space Error' will be inserted into V\$SESSION_WAIT view.

DBA (USER) _RESUMABLE Dictionary View (continued)

Here is a description of the columns in DBA_RESUMABLE:

- SESSION_ID: Session identifier of the statement
- INSTANCE_ID: Instance number of the statement
- COORD_SESSION_ID: Session identifier of Parallel Coordinator
- COORD_INSTANCE_ID: Instance number on which the Parallel Coordinator is running
- SQL_TEXT: The statement; only the first 1000 bytes are stored
- NAME: The name given in the resumable clause of this statement
- STATUS: The status of the statement. Its value can be one of RUNNING, SUSPENDED, COMPLETED, ABORTED, or TIMEOUT.
- ERROR_NUMBER: Error code of the last correctable error occurred. When STATUS is equal to EXECUTING or COMPLETED, its value is set to NULL.
- ERROR_MSG: The error message corresponding to ERROR_NUMBER. It is set to NULL when ERROR_NUMBER is NULL.
- ERROR_PARAMETER1: The 1st parameter for the error message; NULL if no error
- ERROR_PARAMETER2: The 2nd parameter for the error message; NULL if no error
- ERROR_PARAMETER3: The 3rd parameter for the error message; NULL if no error
- ERROR_PARAMETER4: The 4th parameter for the error message; NULL if no error
- START_TIME: The start local time of the statement.
- SUSPEND_TIME: The last local time when the statement was suspended. It is initialized to NULL.
- RESTART_TIME: The last local time when the suspended statement is resumed. It is initialized to NULL.

Note: In parallel execution, if one of the parallel execution server processes encounters a correctable error, that server process suspends its execution. Other parallel execution server processes will continue executing their respective tasks, until either they encounter an error or are directly or indirectly blocked by the suspended server process. When the correctable error is resolved, the suspended process resumes execution and the parallel operation continues execution. If the suspended operation is terminated, the parallel operation aborts, throwing the error to the user. Different parallel execution server processes may encounter one or more correctable errors. This can result in firing an AFTER SUSPEND trigger multiple times, in parallel. Also, if a parallel execution server process encounters a non-correctable error while another parallel execution server process is suspended, the suspended statement is immediately aborted. For parallel execution, every parallel execution coordinator and server process has its own entry in DBA/USER_RESUMABLE view.

Export/Import Enhancements (STATISTICS)

- **Oracle8i uses a combination of ANALYZE and RECALCULATE_STATISTICS parameters.**
- **Oracle9i STATISTICS new Import parameter:**
 - **ALWAYS:** Imports precalculated statistics
 - **SAFE:** Import precalculated statistics only when it is safe
 - **RECALCULATE:** Execute ANALYZE during Import
 - **NONE:** Do not Import or recalculate statistics

ORACLE

2-33

Copyright © Oracle Corporation, 2001. All rights reserved.

Export/Import Enhancements (STATISTICS)

If statistics are requested at export time and analyzer statistics are available for a table, Export places the ANALYZE command to recalculate the statistics for the table into the dump file. In most circumstances, Export will also write the precalculated optimizer statistics for tables, indexes, and columns to the dump file. By default, and unless Import finds certain precomputed statistics as questionable (for example, there are row errors while exporting), Import will always use the precomputed statistics that are found in the export dump file.

In certain situations, the importer might want to import only non-questionable statistics, and may not want to import precomputed statistics (for example, row errors occurred while importing the table). Import now has the STATISTICS parameter. In Oracle8i, you can use a combination of ANALYZE and RECALCULATE_STATISTICS during the import process to effect an equivalent behavior.

The STATISTICS parameter specifies what is done with the database optimizer statistics at import time. It defaults to ALWAYS and can have the following values:

- **ALWAYS:** Always import database optimizer statistics regardless of whether or not they are questionable.
- **NONE:** Do not import or recalculate the database optimizer statistics.
- **SAFE:** Import database optimizer statistics only if they are not questionable. If they are questionable, recalculate the optimizer statistics.
- **RECALCULATE:** Do not import the database optimizer statistics. Instead, recalculate them on import.

Export/Import Enhancements (TABLESPACES Mode)

- Oracle9i new TABLESPACES export parameter exports only tables residing in specified tablespaces.
- Indexes are also exported regardless of their locations.
- You must have EXP_FULL_DATABASE privilege in order to Export in TABLESPACES mode.

```
$ exp system/manager tablespaces=users
```

ORACLE

Export/Import Enhancements (TABLESPACES Mode)

The TABLESPACES parameter specifies that all tables in the tablespace be exported to the Export dump file. This includes all tables contained in the list of tablespaces and all tables that have a partition located in the list of tablespaces. Indexes are exported with their tables, regardless of where the index is stored.

You must have the EXP_FULL_DATABASE role to use TABLESPACES to export all tables in the tablespace.

Note: The TABLESPACES parameter can continue to be used for tablespaces transportation like in previous versions.

Export/Import Enhancements (Resumable Space Allocation)

- **RESUMABLE (y/n):** used to enable and disable resumable space allocation. Default value is **n**
- **RESUMABLE_NAME:** Text string inserted into ***_RESUMABLE** views. When set, it enables Resumable Space Allocation mode for the Export/Import session.
- **RESUMABLE_TIMEOUT:** Specify this parameter in order to change the default timeout period.

ORACLE

2-35

Copyright © Oracle Corporation, 2001. All rights reserved.

Export/Import Enhancements (Resumable Space Allocation)

The **RESUMABLE** parameter enables Resumable Space Allocation Export/Import sessions. Because this parameter is disabled by default, you must set **RESUMABLE=y** in order to use its associated parameters, **RESUMABLE_NAME** and **RESUMABLE_TIMEOUT**.

The **RESUMABLE_NAME** parameter identifies the session that is in Resumable Space Allocation mode. This value is a user-defined text string that is inserted in the **USER[DBA]_RESUMABLE** view to help you identify a specific statement. The default value for this parameter is: **'User USERNAME (USERID), Session SESSIONID, Instance INSTANCEID'**

The **RESUMABLE_TIMEOUT** parameter is used to change the default timeout period of two hours (7200 seconds). The value of the parameter specifies the time period during which an error must be fixed.

Note: If the error is not fixed within the timeout period, execution of the statement is aborted.

Export/Import Enhancements (Flashback)

- **FLASHBACK_SCN**: SCN used to set session snapshot back to
- **FLASHBACK_TIME**: time used to get the SCN closest to the specified time

ORACLE

2-36

Copyright © Oracle Corporation, 2001. All rights reserved.

Export/Import Enhancements (Flashback)

Refer to the beginning of this lesson for more information on the Flashback feature. It is possible to get snapshots of an Export using the above parameters.

Summary

In this lesson, you should have learned about:

- **Two-pass crash recovery**
- **Fast-start time-based recovery limit**
- **Consistent queries as of a specific point in the past using the DBMS_FLASHBACK package**
- **Usage of the DBMS_RESUMABLE package**
- **The new TABLESPACES export mode**

ORACLE

Practice 2-1 Overview

This practice covers the following topics:

- **Use of the DBMS_FLASHBACK package to enable Flashback mode in your session**
- **Modification of the UNDO_RETENTION initialization parameter**
- **Links between SCN and time in Flashback mode**

ORACLE

Practice 2-2 Overview

This practice covers the following topics:

- Use the Resumable Space Allocation feature during a tablespace export or import process
- Create an `AFTER SUSPEND` trigger
- Use the new `RESUMABLE` and `TIMEOUT_RESUMABLE` import parameters

ORACLE

Oracle9i LogMiner Enhancements

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Explain LogMiner new features**
 - DDL statement support
 - Dictionary staleness detection
 - Ability to store dictionary in redo logs
 - Ability to use an online dictionary
 - Supplemental logging of column values
 - Ability to skip log corruptions
- **Describe the LogMiner Viewer**

ORACLE

LogMiner New Features

For use on Oracle9i generated log files:

- **DDL statement support**
- **Ability to translate DML associated with Index Clusters**
- **Support for chained and migrated rows**
- **Direct path inserts**
- **Extract the data dictionary into the redo log files**
- **Ability to use an online dictionary**

ORACLE

LogMiner New Features

- Dictionary staleness detection
- Ability to skip log corruptions
- Generate `SQL_REDO` and `SQL_UNDO` with primary key information

These features available on version 8.0 and higher redo log files:

- View committed transactions only
- Query on actual data values in redo log files

ORACLE

DDL Statement Support

- Previously, DDL statements were mapped to several DML statements on internal tables.
- Starting with Oracle9i, the DDL issued by the user is logged as part of the DDL transaction.
- The OPERATION column of the V\$LOGMNR_CONTENTS shows DDL .
- The SQL_REDO column of the V\$LOGMNR_CONTENTS view shows the actual DDL statement entered.
- DDL support is only for Oracle9i and higher generated log files.

ORACLE

3-5

Copyright © Oracle Corporation, 2001. All rights reserved.

DDL Statement Support

Before Oracle9i, the DDL statements appeared in the log files as a set of DML statements on the internal data dictionary tables. This made these statements nearly impossible to read. As the reader of the log files you would have to infer not only that a DDL statement took place but also what DDL statement it was. Now with Oracle9i, the actual DDL statement is logged in the redo log files. Now LogMiner displays this DDL statement, along with the set of DML statements.

DDL Support does not provide the ability to recover from table drops or truncates since DDL statements do not delete individual rows. DDL support is only for log files created by a Oracle9i or higher database.

DDL Statement Support Example

```
SQL> select username,operation,sql_redo
2  from    v$logmnr_contents
3  where   username = 'HR'
4  and     operation = 'DDL';
```

USERNAME	OPERATION	SQL_REDO
HR	DDL	alter table job_history add final_sal number(8,2)

ORACLE

DDL Statement Support Example

This shows how an ALTER TABLE command would appear in the log. The sql_undo column is NULL for all DDL commands. The multiple DML commands on the data dictionary tables will follow this entry. They are the same entries you would see in the older version of LogMiner.

Data Dictionary Access

- To fully translate the contents of redo logs, LogMiner requires access to a dictionary.
- The procedure `DBMS_LOGMNR_D.BUILD` extracts database dictionary information to:
 - A flat file
 - The redo logs (new in Oracle9i)
- In Oracle9i, it is now possible for LogMiner to use the online data dictionary.

ORACLE

3-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Extracting the Dictionary to a Flat File

The advantage of having the dictionary in a flat file is that it does not consume as many system resources such as it does when it is contained in the redo logs. However, when the dictionary is in a flat file, there is a possibility that the information it contains will be out-of-synchronization with the redo logs being analyzed. The only time a flat file version of the dictionary is required is when the database is unmounted. The directory specified must match `UTL_FILE_DIR` in the parameter file.

```
SQL> EXECUTE dbms_logmnr_d.build -  
      2 (DICTIONARY_FILENAME => 'dictionary.ora' -  
      3 ,DICTIONARY_LOCATION => '/oracle/database' -  
      4 ,OPTIONS => DBMS_LOGMNR_D.STORE_IN_FLAT_FILE);
```

Extracting the Data Dictionary to the Redo Logs

Having the dictionary in the redo logs prevents the problems of inaccurate correlation with the redo logs. The process of extracting the dictionary to the redo logs does consume database resources, but if you limit the extraction to off-peak hours, this should not be a problem. It is also faster than extracting to a flat file.

Using the Online Dictionary

Using the online dictionary as your dictionary ensures that you are using the latest information because LogMiner uses the dictionary currently in use for the database.

Dictionary Information in the Redo Logs

- Starting with Oracle9i, you can have the dictionary information stored in the redo log stream.
- The database must be in archive log mode.
- The database must be open when `DICT_FROM_REDO_LOGS` is used in the `DBMS_LOGMNR.START_LOGMNR` procedure.

```
SQL> execute dbms_logmnr_d.build ( -  
      2  options => dbms_logmnr_d.store_in_redo_logs);
```

ORACLE

3-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Dictionary Information in the Redo Logs

Putting the dictionary information in the log files guarantees a consistent snapshot of the dictionary. Using the redo logs for the dictionary in use with the DDL Tracking option ensures that the dictionary being used by LogMiner is consistent with the data being analyzed. This avoids management of a different set of files, because there is no need to manage the backup of the flat files, as was the case with earlier versions of LogMiner.

The database must be in Archive Log mode to use this option. If it is not, you will receive the following error:

```
ORA-01325: archive log mode must be enabled  
          to build into the logstream.
```

Using an Online Data Dictionary

- The database must be open.
- The user assumes responsibility about dictionary and redo mismatch.
- Specify the `DICT_FROM_ONLINE_CATALOG` option in `dbms_logmnr.start_logmnr`.
- If a dictionary file is specified with this option, it is ignored.
- This feature is useful to analyze recent redo logs.

ORACLE

3-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Using an Online Data Dictionary

In order to activate this feature, you must specify `DICT_FROM_ONLINE_CATALOG` option with the `dbms_logmnr.start_logmnr` procedure.

```
SQL> EXECUTE dbms_logmnr.start_logmnr -  
2 (OPTIONS => DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG);
```

This is useful when you know that the logical structure of the tables have not changed over time and the online data dictionary represents the correct logical structure.

DDL Tracking in the Dictionary

- In Oracle8i, the dictionary snapshot used by LogMiner was static.
- Starting with Oracle9i, LogMiner applies the DDL encountered to its own dictionary.
- LogMiner can translate any objects that were created after the dictionary snapshot was taken.
- Specify the DDL_DICT_TRACKING option in `dbms_logmnr.start_logmnr`.
- The database must be open for this option to be meaningful.
- Only for dictionary in flat file or redo log stream

ORACLE

3-10

Copyright © Oracle Corporation, 2001. All rights reserved.

DDL Tracking in the Dictionary

In Oracle8i, the dictionary snapshot used by LogMiner was *static*. In other words, LogMiner lacked the ability to apply the DDL operations it encountered in the set of redo log records being analyzed, to its own dictionary. As a result, LogMiner could not translate any objects that were created after the snapshot was taken.

From Oracle9i, LogMiner keeps track of the DDL operations in the input log stream (the set of logs you are processing) and updates its internal dictionary appropriately. If you are using a flat file version of the LogMiner dictionary or a version in the redo logs, then the dictionary is updated appropriately provided you specify the DDL_DICT_TRACKING option when you start LogMiner.

Another option called NO_DICT_RESET_ONSELECT can be specified together with DDL_DICT_TRACKING. It prevents LogMiner from reloading its dictionary at the beginning of each select operation on V\$LOGMNR_CONTENTS. This can be an advantage because it can potentially be very time consuming to refresh the dictionary if a DDL operation has updated the internal LogMiner dictionary. However, if you use this option, you may get incorrect SQL_REDO and SQL_UNDO information for objects that are modified in the redo log files because the dictionary has not been refreshed.

Note: Multiple options can be concatenated using the Addition symbol (+). For example, to use the dictionary information that was created in the redo log stream and to enable DDL tracking the statement would look like:

```
SQL> EXECUTE dbms_logmnr.start_logmnr -  
2 (OPTIONS => DBMS_LOGMNR.DICT_FROM_REDO_LOGS -  
3 + DBMS_LOGMNR.DDL_DICT_TRACKING );
```

Oracle9i: New Features for Administrators 3-10

Dictionary Staleness Detection

- Previously, LogMiner could not ascertain if the dictionary snapshot used was synchronized with the log files being analyzed.
- In Oracle9i, versioning information is logged with each object.
- The LogMiner dictionary is aware of different object versions.
- LogMiner can now verify that the dictionary is synchronized with the log stream being processed.

ORACLE

3-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Staleness

LogMiner needs to know that any particular object in the redo logs really matches the one in its dictionary file. This is facilitated by the use of a new internal object version number in Oracle9i. This number is incremented each time an object is changed or re-created. In this way LogMiner can determine whether the dictionary it is using is in *sync* or *stale* in reference to the object being analyzed in the log. This all happens automatically to ensure that the results you see in the LogMiner views are accurate.

Skip Past Log Corruptions

- By default, LogMiner stops when a log corruption is encountered.
- Users can now specify that LogMiner should attempt to proceed beyond the corruption.
- This is done with a new option for the procedure `dbms_logmnr.start_logmnr` called `SKIP_CORRUPTION`.
- LogMiner indicates how many blocks have been skipped so far.

ORACLE

3-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Skip Past Log Corruptions

Any corruptions in the redo log are skipped during select operations from the `V$LOGMNR_CONTENTS` view. The rows that are retrieved after a corruption are flagged. In `V$LOGMNR_CONTENTS`, the `INFO` column displays the text “Log File Corruption Encountered.” Also, for every corrupt redo record encountered, an informational row, indicating how many blocks have been skipped, is displayed.

The default is for the select operation to terminate at the first corruption it encounters in the log file.

Display Only Committed Transactions Information

- Upon request, only information from committed transactions is displayed.
- This is done with a new option for the procedure `dbms_logmnr.start_logmnr` called `COMMITTED_DATA_ONLY`.
- Omits information regarding in-progress
- A committed transaction is defined as a set of statements that end with a commit or rollback.

ORACLE

3-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Display Only Committed Transactions Information

In order to activate this feature, you should specify the `COMMITTED_DATA_ONLY` option with the `dbms_logmnr.start_logmnr` procedure.

```
SQL> EXECUTE dbms_logmnr.start_logmnr( -  
      2  OPTIONS => DBMS_LOGMNR.COMMITTED_DATA_ONLY );
```

When this option is set, rows returned from `V$LOGMNR_CONTENTS` are grouped together by transaction identifier, and only rows belonging to a committed transaction are returned. Transactions are returned in commit SCN order.

In the initial release of this feature of LogMiner, the committed data is defined as a set of statements in a transaction. Therefore, a set of statements that have been rolled back will be seen in the output. This is because Oracle treats a rolled back transaction as a committed transaction that had no effect on the data.

Primary Key Information

- **Default format of the entries in the SQL_REDO and SQL_UNDO columns of V\$LOGMNR_CONTENTS:**

```
update hr.employees set salary = 5500 where
salary = 4700 and rowid = 'AAABEpAADAAAAAmAAA'
```

- **Use supplemental log groups at the table or database level to get the PK (or any other columns) in the WHERE clause of the statement**
- **You still get the ROWID pseudo column predicate**
- **This increases the amount of redo, and if done at the database level, it can be very significant.**

ORACLE

3-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Primary Key Information

By default, the entries in the SQL_UNDO and SQL_REDO columns show the ROWID for a predicate, along with any others, for the statement that was entered. This works well if the statement is to be applied to the same database that the statement was issued on, and if the ROWID has not changed. A ROWID can change with an ALTER TABLE <table name> MOVE command, an export and import, or the row being deleted and reinserted.

If you want to see the Primary Key or any other columns that would uniquely identify the row, you must set up a supplemental redo log group. The command does not check to see if the columns you have set up are indeed a Primary Key or Unique Key.

For example, if the following statement were issued:

```
SQL> ALTER TABLE hr.employees
2 ADD SUPPLEMENTAL LOG GROUP emp_group_1
3 (last_name, employee_id);
```

then for all of updates to the hr.employee table the old values of last_name, employee_id for the row being updated will be logged. In this case the update above would show up in the SQL_REDO column as:

```
UPDATE hr.employee SET salary = 5500 WHERE salary = 4700 AND
last_name = 'King' AND employee_id = '4300' AND rowid =
'AAABEpAADAAAAAmAAA';
```

Primary Key Information (continued)

This can also be done at the database level with:

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA  
      2 (PRIMARY KEY) COLUMNS;
```

Note: if this is done and a table doesn't have a primary key, then all columns for the table are logged.

LogMiner Restrictions

- **The following are not supported:**
 - Data types LONG and LOB
 - Object types
 - Collections (nested tables and VARRAYs)
 - Object Refs
 - Index Organized Tables (IOTs)
- **For direct path, LOGGING must be enabled and in ARCHIVELOG mode**

ORACLE

3-16

Copyright © Oracle Corporation, 2001. All rights reserved.

LogMiner Restrictions

If LogMiner sees any of the mentioned data types or segments, it will generate invalid SQL in the SQL_REDO and SQL_UNDO columns of V\$LOGMNR_CONTENTS. However, all the other data in V\$LOGMNR_CONTENTS will still be valid.

Note: LogMiner can be run in a shared server configuration, as long as the session is connected with a dedicated server. However, this is not recommended since the resources used by LogMiner could severely impact the performance of the other sessions connected through shared server.

LogMiner Views

- V\$LOGMNR_CONTENTS
- V\$LOGMNR_DICTIONARY
- V\$LOGMNR_LOGS
- V\$LOGMNR_PARAMETERS

ORACLE

3-17

Copyright © Oracle Corporation, 2001. All rights reserved.

LogMiner Views

All of these views only contain data when LogMiner is started:

V\$LOGMNR_CONTENTS shows changes as stored in the log files.

V\$LOGMNR_DICTIONARY shows information about the LogMiner dictionary file, provided the dictionary was created using the STORE_IN_FLAT_FILE option; otherwise this view is empty.

V\$LOGMNR_LOGS shows information about specified log files.

V\$LOGMNR_PARAMETERS shows information about optional LogMiner parameters, including starting and ending SCNs and starting and ending times.

LogMiner Viewer

- **Provides a GUI for LogMiner**
- **Facilitates query building in the redo log files**
- **Stores queries for subsequent use**
- **Returns rows with the primary key displayed**
- **Displays content-aware changes**
- **Optionally, it displays only committed transactions**
- **Requires Enterprise Manager**

ORACLE

3-18

Copyright © Oracle Corporation, 2001. All rights reserved.

LogMiner Viewer

Specifying queries using either the command-line interface or the new Oracle9i LogMiner Viewer is easier because:

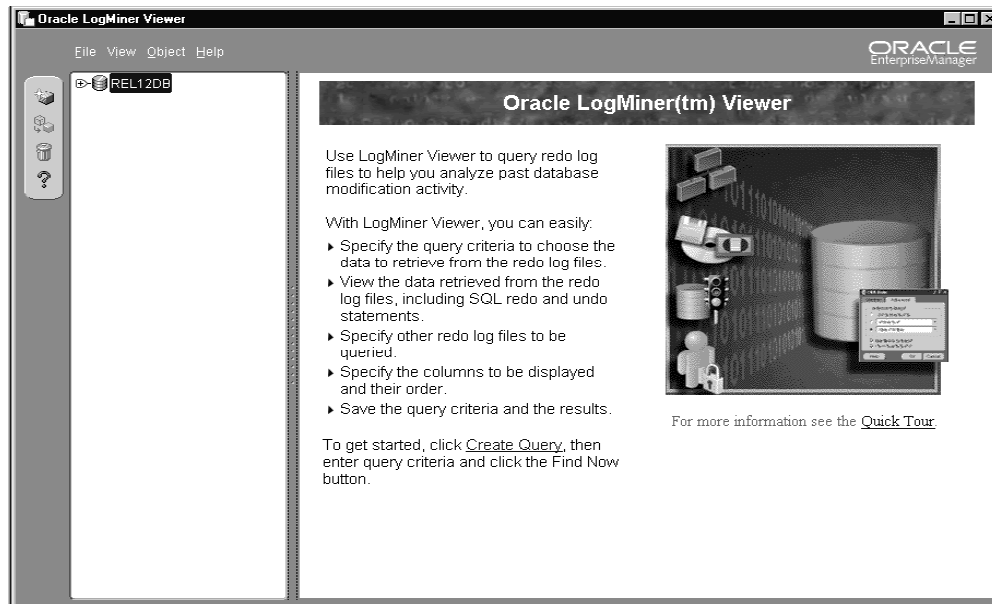
- Rows are returned with the primary key displayed.
- Changes can be requested by the content of a change, for instance, all changes to an employee named Bartholomew.
- Queries can optionally return only committed transactions.

Oracle9i LogMiner Viewer adds a GUI-based interface to the pre-existing command-line interface (CLI), which:

- Makes the LogMiner setup process easier
- Displays the results of queries in a readable form with a minimum of work from the user
- Tracks queries and their results, making it easier to skip to prior queries and re-execute.

The completeness of Oracle9i LogMiner not only means increased productivity with less time to learn and use; but also that it can be used with confidence in all application environments.

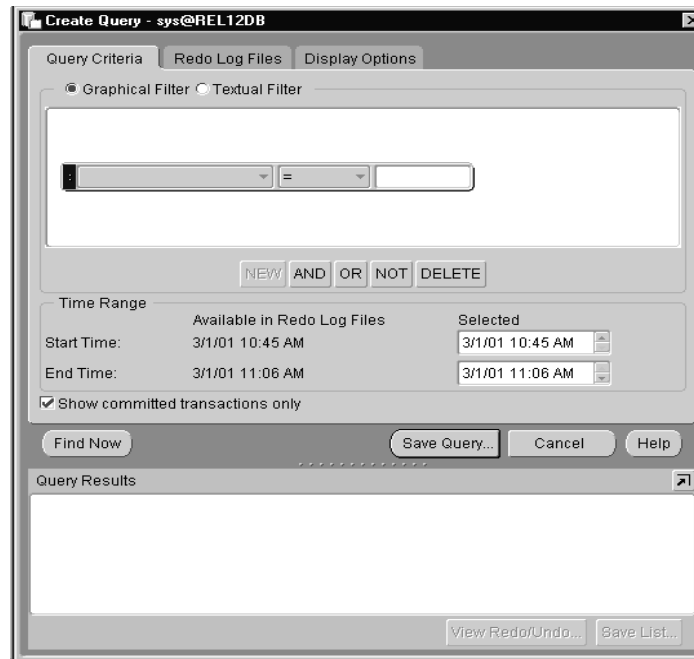
LogMiner Viewer



LogMiner Viewer (continued)

The main LogMiner Viewer window shows the navigator tree on the left and the introductory detail panel on the right. The tree contains a list of any discovered databases.

LogMiner Viewer

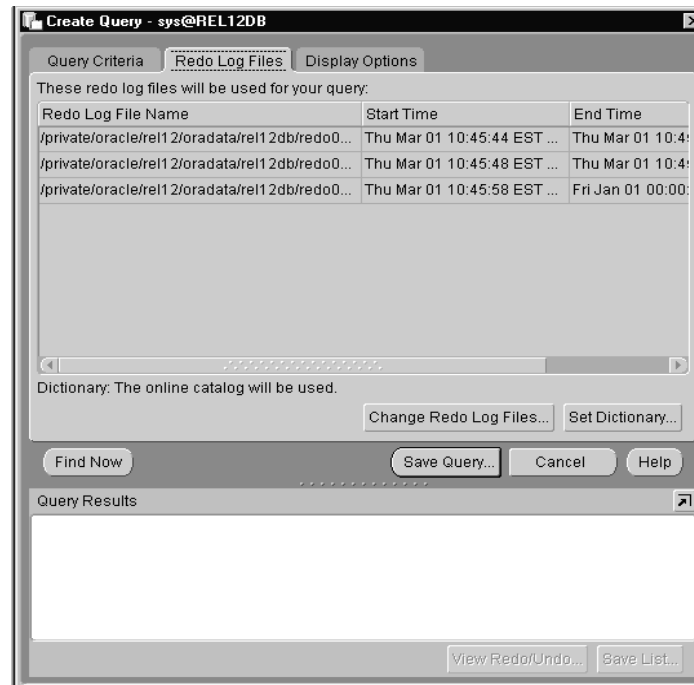


ORACLE

LogMiner Viewer (continued)

The Query Criteria tabbed page displays dialog for creating a query into the redo log files, initial property page. In this region, the user can specify query criteria such as time range, table name, table owner, SQL operation and so on.

LogMiner Viewer

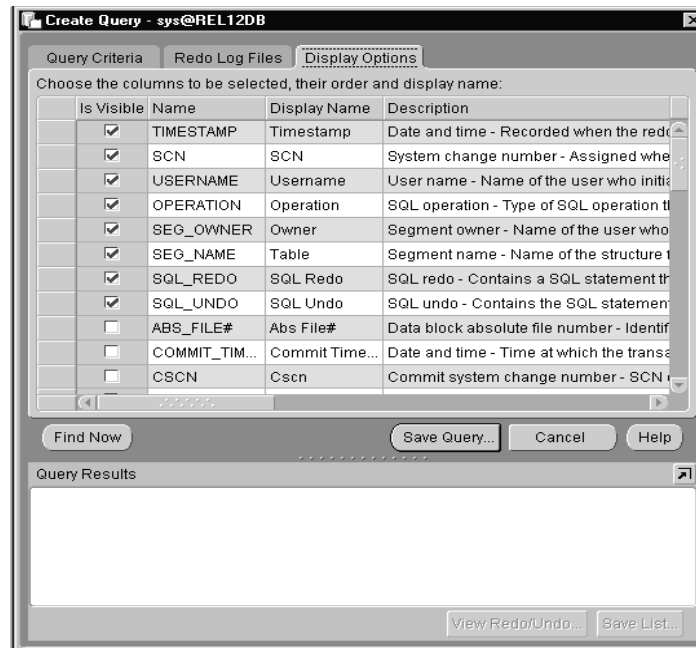


ORACLE

LogMiner Viewer (continued)

The Redo Log Files tabbed page shows dialog for creating a query into the redo log files, second property page. This region displays the set of redo log files used by default, which are the online redo log files for the database currently selected in the tree. The user can change the redo log files here.

LogMiner Viewer



ORACLE

LogMiner Viewer (continued)

The Display Options tabbed sheet page shows dialog for creating a query into the redo log files, third property page. This region displays the columns of redo log file data that will be displayed in the query results. These columns come from the internal LogMiner table: V\$LOGMNR_CONTENTS. The user can select the columns to be displayed and can modify the name used for the column header.

LogMiner Viewer

Create Query - sys@REL12DB

Query Criteria | Redo Log Files | Display Options

☒ Graphical Filter ☐ Textual Filter

Table = EMPLOYEES

NEW AND OR NOT DELETE

Time Range

Available in Redo Log Files

Start Time: 3/1/01 10:45 AM

End Time: 3/1/01 11:06 AM

Selected

3/1/01 10:45 AM

3/1/01 11:06 AM

☒ Show committed transactions only

Find Now Save Query... Cancel Help

Query Results (March 1, 2001 11:09:23 AM EST)

Timesta...	SCN	Userna...	Operation	Owner	Table	S
01-Mar-200...	206774	SYS	UPDATE	HR	EMPLOYEES	up
01-Mar-200...	206776	SYS	UPDATE	HR	EMPLOYEES	up
01-Mar-200...	206778	SYS	UPDATE	HR	EMPLOYEES	up
01-Mar-200...	206780	SYS	UPDATE	HR	EMPLOYEES	up
01-Mar-200...	206782	SYS	UPDATE	HR	EMPLOYEES	up

View Redo/Undo... Save List...

ORACLE

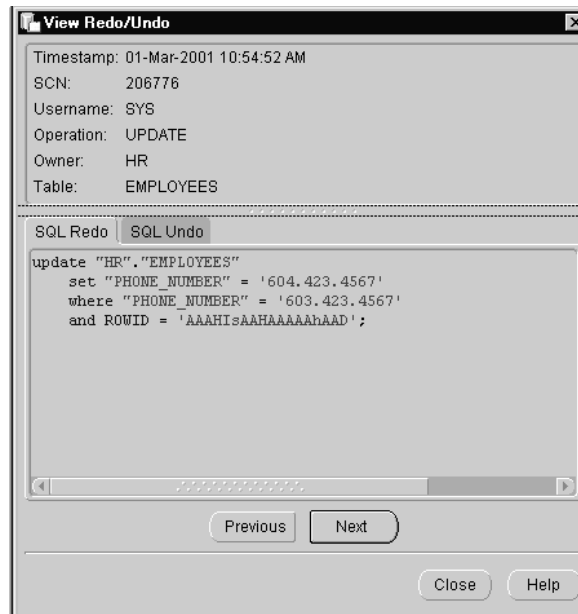
3-23

Copyright © Oracle Corporation, 2001. All rights reserved.

LogMiner Viewer (continued)

After clicking the Find Now button, the lower half of the Query Criteria tabbed page displays the results.

LogMiner Viewer



ORACLE

LogMiner Viewer (continued)

The user can select a row in the query results table and click the View Redo/Undo button to see an alternate view of the information from that row, including the SQL redo and SQL undo data.

Summary

In this lesson, you should have learned how to:

- **Use some Oracle9i LogMiner new features:**
 - DDL statement support
 - Dictionary staleness detection
 - The online dictionary
 - Skipping log corruptions
- **Use the Oracle9i LogMiner Viewer**

ORACLE

Practice 3-1 Overview

This practice covers the following topics:

- **Create a dictionary in a flat file**
- **Create a table after the dictionary file is created**
- **Perform DDL and DML commands on the table**
- **Use LogMiner to see the commands on the table**

ORACLE

4

Backup and Recovery

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Describe new RMAN manageability features**
 - New backup enhancements
 - New restore and recovery enhancements
- **Describe new RMAN reliability features**
 - Block media recovery (BMR)
- **Trial Recovery**
- **Describe other miscellaneous RMAN improvements**

ORACLE

RMAN Manageability Enhancements

- **Persistent configuration parameters**
 - Retention policies
 - Automatic channel allocation
 - Other configuration commands
- **Backup and restore enhancements**
- **New Enterprise Manager Interface**

ORACLE

RMAN Manageability Enhancements

The main goal of Oracle9i Recovery Manager is to improve the ability to manage backups and recoveries, with configuration of user preferences when RMAN operations are run, file optimization during backup and restore, and enhanced automated failover.

Persistent Configuration Parameters

- Customizable configuration parameters simplify RMAN operations.
- Default settings are set once and used for subsequent jobs.
- A DBA can invoke RMAN and back up a database with one command: `BACKUP DATABASE`.
- Oracle9i provides the new `CONFIGURE` command to override default settings persistently.
- The configuration values are stored in the control file and are resynchronized to the recovery catalog as necessary.

ORACLE

4-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Persistent Configuration Parameters

To see the configurable parameters, use the `show all` command. The following is a list of the parameters and their defaults:

```
RETENTION POLICY TO REDUNDANCY 1
BACKUP OPTIMIZATION OFF
DEFAULT DEVICE TYPE TO DISK
CONTROLFILE AUTOBACKUP OFF
CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'
DEVICE TYPE DISK PARALLELISM 1
DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1
ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1
MAXSETSIZE TO UNLIMITED
SNAPSHOT CONTROLFILE NAME TO
'<oracle_home>/dbs/snapcf_<db_name>.f'
```

EXCLUDE, CHANNEL, AUXNAME, and AUXILIARY CHANNEL do not have default values.

Note: Some parameters are platform dependent; check your system for your actual defaults.

Retention Policies

A retention policy describes which backups will be kept and for how long.

There are two retention policy types:

- **Recovery window:** establishes a period of time within which point-in-time recovery must be possible
- **Redundancy:** establishes a fixed number of backups that must be kept. Backups in excess of this can be deleted.
- **These policies are mutually exclusive and can be set with the CONFIGURE command.**

The default retention policy is REDUNDANCY 1.

ORACLE

4-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Retention Policies

A retention policy is a user's description of which backups are kept and for how long. The value of the retention policy is set by the new CONFIGURE command. The best practice is to establish a period of time during which it is possible to discover logical errors and fix the affected objects by doing a point-in-time recovery to just before the error occurred. This period of time is called the recovery window. This policy is specified in number of days.

For each data file, there must always exist one backup which satisfies the following condition:

```
SYSDATE-CHECKPOINT_TIME >= recovery_window
```

For example, if the policy were set as follows:

```
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS;
```

Then for each file there must be a backup that satisfies:

```
SYSDATE - (SELECT CHECKPOINT_TIME FROM V$DATAFILE) >= 7
```

Although the recovery window is the best implementation for specifying how long backups should be kept, the number of backups that must be stored as defined by the recovery window is not constant. For DBAs who require the exact number of backups, they may set the retention policy based upon the redundancy option. This option requires a specified number of backups to be cataloged before any backup is identified as obsolete. This policy is specified as number of backups.

CONFIGURE RETENTION POLICY

```
RMAN> CONFIGURE RETENTION POLICY  
      TO RECOVERY WINDOW OF 5 DAYS;
```

```
RMAN> CONFIGURE RETENTION POLICY  
      TO REDUNDANCY 5;
```

```
RMAN> CONFIGURE RETENTION POLICY TO NONE;
```

Use the DELETE OBSOLETE command to delete backups or copies that are no longer required based on the chosen retention policy.

ORACLE

4-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Retention Policies (continued)

The parameter `CONTROL_FILE_RECORD_KEEP_TIME` determines when the records or entries in the control file will be reused with new information. When the data is replaced, RMAN is no longer able to reconstruct a list of backup files required to do a valid restore. If the value specified for the recovery window is greater than the `CONTROL_FILE_RECORD_KEEP_TIME` parameter, a recovery catalog is required.

The `DELETE OBSOLETE` command is used to delete obsolete backup sets based on the retention policy. Backup sets that are identified to never expire are not affected by the retention policy or `DELETE OBSOLETE` command.

There are several data dictionary views that can be used to see configuration settings: `V$RMAN_CONFIGURATION`, `RC_RMAN_CONFIGURATION`, `RC_TABLESPACE`, and `RC_DATAFILE`.

Note: RMAN cannot implement an automatic retention policy if backups are deleted using non-RMAN methods. One such method is the media manager's tape retention policy. Ideally, the media manager will never expire a tape until all RMAN backups residing on that tape have been removed from the media manager's repository.

Note: There is a difference between `CONFIGURE RETENTION POLICY TO NONE` and `CONFIGURE RETENTION POLICY CLEAR`. The first means that there is no retention policy: backups will never expire, and `DELETE OBSOLETE` will give an error. The second means that the default retention policy (`REDUNDANCY 1`) will be in effect.

Automatic Channel Allocation

- **This feature applies to BACKUP, COPY, and RESTORE commands.**
- **A channel is automatically allocated if one is not explicitly specified in the RMAN command.**
- **Default values are specified with the CONFIGURE command.**
- **The benefit of this feature is a usability improvement due to the simplification of subsequent commands.**

ORACLE

4-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Automatic Channel Allocation

When a channel is automatically allocated by RMAN, its name is in the format `ora_devicetype_n` (this is, `ora_sbt_tape_n` or `ora_disk_n`).

RMAN knows the command that is being run. For backups, only a single type of channel is allocated. For restores, RMAN knows which device types are required, allocating all necessary channels.

CONFIGURE CHANNEL

Used to configure settings that will be used for the channels (all or individual):

```
CONFIGURE CHANNEL [n] <channel_option_list>;

channel_option_list := TYPE, NAME, PARMS,
                      CONNECT STRING, DEBUG,
                      FORMAT, TRACE,
                      MAXPIECESIZE, RATE,
                      MAXOPENFILES, SEND
```

ORACLE

4-8

Copyright © Oracle Corporation, 2001. All rights reserved.

CONFIGURE CHANNEL

Within I/O channel allocation, additional properties of a channel may be specified. The `CONFIGURE CHANNEL` command gives the DBA the capability to automatically substitute parameter option values such as `NAME`, `PARMS`, `CONNECT STRING`, `DEBUG`, `FORMAT`, `TRACE`, `MAXPIECESIZE`, `RATE`, `MAXOPENFILES`, and `SEND` to be used when I/O is needed.

When parallelizing channels, it may be necessary for each channel to be uniquely identified. A typical example of this is in a Real Application Clusters architecture. In the Real Application Clusters environment, online redo logs are archived to different nodes in a cluster. It is necessary for RMAN to connect to each node to backup the archived logs.

In this example the channel is set to make backups in a particular directory:

```
RMAN> CONFIGURE CHANNEL 1 DEVICE TYPE DISK FORMAT
      '/databases/ed27/data/backup/ed27_%U' ;
```

CONFIGURE CHANNEL (continued)

The following substitution variables are available in `FORMAT` strings to aid in generating unique filenames. The formatting of this information varies by platform.

Parameter	Description
%c	Specifies the copy number of the backup piece within a set of duplexed backup pieces. If you did not duplex a backup, then this variable is 1 for backup sets and 0 for proxy copies. If one of these commands is enabled, then the variable shows the copy number. The maximum value for %c is 256.
%d	Specifies the name of the database
%D	Specifies the current day of the month from the Gregorian calendar in format DD
%F	Combines the DBID, day, month, year, and sequence into a unique and repeatable generated name
%M	Specifies the month in the Gregorian calendar in format MM
%n	Specifies the name of the database, padded on the right with x characters to a total length of eight characters. For example, if the prod1 is the database name, then the padded name is prod1xxx.
%p	Specifies the piece number within the backup set. This value starts at 1 for each backup set and is incremented by 1 as each backup piece is created. Note: If you specify PROXY, then the %p variable must be included in the <code>FORMAT</code> string either explicitly or implicitly within %U.
%s	Specifies the backup set number. This number is a counter in the control file that is incremented for each backup set. The counter value starts at 1 and is unique for the lifetime of the control file. If you restore a backup control file, then duplicate values can result. Also, <code>CREATE CONTROLFILE</code> initializes the counter back to 1.
%t	Specifies the backup set time stamp, which is a 4-byte value derived as the number of seconds elapsed since a fixed reference time. The combination of %s and %t can be used to form a unique name for the backup set.
%T	Specifies the year, month, and day in this format: YYYYMMDD
%u	Specifies an 8-character name constituted by compressed representations of the backup set number and the time the backup set was created.
%U	Specifies a convenient shorthand for %u_%p_%c that guarantees uniqueness in generated backup filenames. If you do not specify a format, RMAN uses %U by default.
%Y	Specifies the year in this format: YYYY
%%	Specifies the % character. For example, %%Y translates to the string %Y.

CONFIGURE CHANNEL

- If the channel number [*n*] is specified, the settings apply only to the channel.
- If not, the settings apply to all channels of the specified type.
- The device type option is always required.
- To remove the options for a specific channel or device type, use:

```
CONFIGURE CHANNEL device_type_spec [n] CLEAR
```

ORACLE

CONFIGURE CHANNEL (continued)

Examples of settings for all channels:

```
CONFIGURE CHANNEL TYPE SBT;  
CONFIGURE CHANNEL DEVICE TYPE sbt MAXPIECESIZE 1G;  
CONFIGURE CHANNEL DEVICE TYPE sbt RATE 1700K;
```

Examples of settings for a specific channel:

```
CONFIGURE CHANNEL 1 TYPE SBT CONNECT 'node1';  
CONFIGURE CHANNEL 2 TYPE SBT CONNECT 'node2';
```


CONFIGURE DEVICE TYPE . . . PARALLELISM

- Determines how many channels of the specified type are allocated for automated backups
- Examples:

```
RMAN> CONFIGURE DEVICE TYPE SBT PARALLELISM 2;
```

```
RMAN> CONFIGURE DEVICE TYPE DISK PARALLELISM 3;
```

ORACLE

4-11

Copyright © Oracle Corporation, 2001. All rights reserved.

CONFIGURE DEVICE TYPE . . . PARALLELISM

The maximum degree of parallelism may be limited by your Media Management Library (MML) provider. Please refer to your MML documentation for details.

Note: For tape (TYPE SBT), the degree of parallelism should be set to the number of physical tape drives. Restore degradation will occur otherwise.

CONFIGURE DEFAULT DEVICE TYPE

- Specifies which device type is used for automated backups
- Valid device type values are: DISK or SBT.
- The default is DISK.
- Using CLEAR sets it back to the default.

```
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO SBT ;
```

```
RMAN> CONFIGURE DEFAULT DEVICE TYPE CLEAR ;
```

ORACLE

CONFIGURE BACKUP COPIES

- It specifies how many identical copies of each backup are created.
- The number of copies for data files, and archive logs can be set independently.
- Range is 1 to 4; 1 is the default.
- Device type is DISK or SBT.

```
RMAN> CONFIGURE DATAFILE BACKUP COPIES  
      FOR DEVICE TYPE DISK TO 2;
```

```
RMAN> CONFIGURE ARCHIVELOG BACKUP COPIES  
      FOR DEVICE TYPE SBT TO 3;
```

ORACLE

CONFIGURE BACKUP COPIES

The SET BACKUP COPIES command can be used to override CONFIGURE BACKUP COPIES for an individual job.

CONFIGURE EXCLUDE

- Useful to prevent repeated backups of read-only or offlined tablespaces
- The system tablespace may not be excluded.
- Explicit backup commands can override the exclusion.
- The exclusion is an attribute of the tablespace so even files added in the future will be affected.
- Use the `CLEAR` option to include the tablespace in backups again.

```
RMAN> CONFIGURE EXCLUDE FOR TABLESPACE test;
```

```
RMAN> CONFIGURE EXCLUDE FOR TABLESPACE test CLEAR;
```

ORACLE

CONFIGURE EXCLUDE

Using an explicit backup command, for example, `BACKUP TABLESPACE`, will ignore the exclusion setting.

CONFIGURE SNAPSHOT CONTROLFILE and CONFIGURE AUXNAME

For consistency, two commands have been renamed:

- **SET SNAPSHOT CONTROLFILE** has been renamed to **CONFIGURE SNAPSHOT CONTROLFILE**:

```
RMAN> CONFIGURE SNAPSHOT CONTROLFILE  
NAME TO '/tmp/snapcf_c82.f';
```

- **SET AUXNAME** has been renamed to **CONFIGURE AUXNAME**:

```
RMAN> CONFIGURE AUXNAME FOR  
DATAFILE 2 TO '/tmp/df2.dbf';
```

The old syntax is still supported for backward compatibility.

ORACLE

CONFIGURE SNAPSHOT CONTROLFILE and CONFIGURE AUXNAME

When RMAN needs to resynchronize from a read-consistent version of the control file, it creates a temporary snapshot control file. RMAN needs a snapshot control file only when resynchronizing with the recovery catalog or when making a backup of the current control file.

The default value for the snapshot control file is platform-specific and depends on the Oracle home. For example, the default filename on some UNIX platforms in Oracle9i is `$ORACLE_HOME/dbs/snapcf_@.f`.

In general, you should need to set the control file location only in these situations:

- You are running RMAN in an Real Application Clusters configuration and need a snapshot control file accessible by each node.
- You are upgrading to the current release from a pre-8.1.7 release. In pre-8.1.7 releases, the default location for the snapshot control file was not dependent on the Oracle home, whereas in the current release the default location is dependent on the Oracle home.

Auxiliary filenames can be configured for specified target datafiles. For example, you can set the auxiliary name for datafile number 2 to `/df2.f`, and then unspecify this auxiliary name by running `CONFIGURE AUXNAME FOR DATAFILE 2 NULL`.

If you are performing tablespace point-in-time recovery or using the `DUPLICATE` command, setting `AUXNAME` allows you to pre-configure the filenames for use on the auxiliary database without manually specifying the auxiliary filenames during the procedure.

CONFIGURE CONTROLFILE AUTOBACKUP

Setting CONFIGURE CONTROLFILE AUTOBACKUP to ON, RMAN automatically backs up the control file:

- **After every BACKUP or COPY command issued at the RMAN prompt**
- **Whenever a BACKUP or COPY command within a RUN block is followed by a command that is neither BACKUP nor COPY**
- **At the end of every RUN block if the last command in the block was either BACKUP or COPY.**

Default is OFF.

ORACLE

4-16

Copyright © Oracle Corporation, 2001. All rights reserved.

CONFIGURE CONTROLFILE AUTOBACKUP

The purpose of the control file autobackup is to provide a way to restore the backup repository contained in the control file when the control file is lost and the recovery catalog is either lost or was never used. You do not need a recovery catalog or target control file to restore the control file autobackup. For example, you can issue:

```
RESTORE CONTROLFILE FROM AUTOBACKUP;
```

By default, the control file autobackup feature uses a well-known filename format so that RMAN can restore it without a repository. The mandatory %F substitution variable is defined for this purpose and has the following format:

```
c-FFFFFFFF-YYYYMMDD-QQ
```

where:

- **FFFFFFFF** represents the database identifier. The dbid is printed in decimal format so that it can be easily associated with the database.
- **YYYYMMDD** represents a time stamp in the Gregorian calendar of the day the backup is generated.
- **QQ** represents the sequence in hexadecimal number that starts with 00 and has a maximum of FF (256).

CONFIGURE CONTROLFILE AUTOBACKUP (continued)

If the channel used for the control file autobackup is of type DISK, then the name of the control file backup is operating system specific. For example, on some UNIX platforms it defaults to \$ORACLE_HOME/dbs/%F.

If you allocate a channel to a tape device, then the format defaults to %F.

You can use the CONFIGURE command to change the default autobackup format to a new value. Note that you must include the %F format in the string, or RMAN signals an error. This format change is persistent across all RMAN sessions.

For example:

```
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO  
'/databases/ed27/data/backup/%F.bck';
```

Long Term Backups

- **Backups can be archived for much longer than the time indicated by the retention policy.**
- **The `KEEP` option provides this functionality.**
- **These backups are kept in the recovery catalog without affecting the retention policy:**

```

RMAN> BACKUP ... KEEP
      [UNTIL TIME 'date' | FOREVER] [NOLOGS | LOGS];

```

- **`NOLOGS` option is not valid for online backups.**
- **The default is `NOKEEP`.**

ORACLE

Long Term Backups

`KEEP` may be specified either on the `BACKUP` or `COPY` commands, when creating a new backup, or on the `CHANGE` command, to alter the retention time or recoverability for an existing backup.

`UNTIL TIME` specifies the date until which the backup must be kept.

`FOREVER` specifies that the backup never expires. A recovery catalog is required to be used when `FOREVER` is specified, otherwise the backup records will eventually age out of the controlfile. RMAN will signal an error if you try to backup with `KEEP FOREVER` without a recovery catalog.

`LOGS` specifies that it is possible to recover this backup to any point in time. All the required archive logs must remain available as long as this backup is available. This is the default.

`NOLOGS` specifies that it will not be possible to recover this backup. The only use for this backup is to restore the database to the point in time that the database was taken. The archived logs required to recover this backup will not be kept. This option is not valid for online backups.

`NOKEEP` specifies that the backup expires according to the user's retention policy. This is the default unless the `KEEP` option is specified.

Mirrored Backups

- This functionality already exists with the **SET DUPLEX** command.
- New features include:
 - A default duplex setting
 - A multi-valued format option for duplexed backups to disk

```
RUN {  
  SET BACKUP COPIES 2;  
  BACKUP DATABASE  
  FORMAT '/dir1/%U', '/dir2/%U';  
}
```

ORACLE

4-19

Copyright © Oracle Corporation, 2001. All rights reserved.

Mirrored Backups

It is possible to specify up to four values for the format option. The second, third, and fourth values are used when set backup copies is in effect.

When choosing which format value to use for each backup piece, RMAN will use the first format value for copy number 1, the second format value for copy number 2, and so on. If the number of format values exceeds the number of copies, the extra formats will not be used. If the number of format values is less than the number of copies, the format values will be reused, starting with the first one. This is illustrated in the following examples:

```
SET BACKUP COPIES 4;  
BACKUP DATABASE FORMAT '/dir1/%U', '/dir2/%U';
```

In this example, the first copy of each backup piece will be placed in dir1, the second in dir2, the third in dir1, and the fourth in dir2.

```
SET BACKUP COPIES 2;  
BACKUP DATABASE FORMAT  
'/dir1/%U', '/dir2/%U', '/dir3/%U', '/dir4/%U';
```

In this example, the first copy of each backup piece will be placed in dir1 and the second in dir2. dir3 and dir4 will not be used.

Note: The RUN command and the braces '{' and '}' are required for this command.

Backup File Optimization

- Useful in preventing repeated backups of read-only tablespaces or archive logs.
- If the header of the backed up file matches the online file header, the file is ignored by new backup commands.
- One exception is when the existing backup is older than the recovery window.
- The `CONFIGURE` command sets this feature:

```
RMAN> CONFIGURE BACKUP OPTIMIZATION [ON|OFF|CLEAR];
```

- Use the `FORCE` option of the `BACKUP` command to bypass this optimization.

ORACLE

4-20

Copyright © Oracle Corporation, 2001. All rights reserved.

Backup File Optimization

If RMAN is about to back up a file, and the file has already been backed up to the same device type, RMAN skips the backup of that file.

For archive logs, same file means same dbid, thread, sequence, and resetlogs data. For datafiles, same file means same dbid, checkpoint, and resetlogs data. For backup sets, same file means the specified backup set has already been backed up to the specified device type.

RMAN does not signal an error if this optimization causes all the files to be skipped.

If `DELETE INPUT` option is used, RMAN deletes all files that would be backed up, even if some of those are not included due to optimization.

If the user's retention policy is set to recovery window, RMAN always backs up any files whose most recent backup is older than the recovery window.

Note: Use caution when enabling backup optimization if you use a media manager that has an expiration policy. The media manager can expire tapes containing backups, and RMAN will not make new backups because of optimization. Run `CROSSCHECK` periodically to synchronize the repository with the media manager.

Restartable Backups

- Unsuccessful backups can now be restarted.
- Only missing or incomplete files are backed up based on backup time.
- Files that were successfully backed up before the interruption are skipped.
- The new option `NOT BACKED UP [SINCE TIME 'timestamp']` must be specified to restart a failed backup.

```
RMAN> BACKUP DATABASE NOT BACKED UP  
      SINCE TIME '12-OCT-00 13:00:00';
```

ORACLE

Restartable Backups

It is inefficient and unnecessary to backup files that have not changed since the last backup.

If a DBA wants to use this feature, the `NOT BACKED UP [SINCE TIME]` option must be used. This instructs RMAN to backup files that have not been backed up after a specified time stamp.

If the `SINCE TIME` option is not specified, only those files that have never been backed up will be done this time. This is useful for taking a backup of new files immediately after they have been created.

The `timestamp` should be either a date in the current `NLS_DATE_FORMAT` or a SQL date expression like `'sysdate - 1'`.

When determining whether a file has been backed up or not, the date entered in the command is compared with the completion time of the most recent backup. The completion time for a file in a backup set is the completion time of the entire backup set; in other words, all files in the same backup set have the same completion time.

Note: This feature only works on Oracle9i backups. It will not restart backups taken with previous releases of the Oracle server.

Archive Log Backup

- Archive logs which have not been backed up can now be included with a data file backup.
- The **PLUS ARCHIVELOG** option of the **BACKUP** command is used.
- **PLUS ARCHIVELOG** is the default when backing up to tape:

```
RMAN> BACKUP DATAFILE 3 PLUS ARCHIVELOG;
```

- **RMAN** does not signal an error if the **BACKUP ARCHIVELOG ALL** command finds no logs to backup.

ORACLE

4-22

Copyright © Oracle Corporation, 2001. All rights reserved.

Archive Log Backup

RMAN does not signal an error if the **BACKUP ARCHIVELOG ALL** command finds no logs to back up. This probably means that no new logs were generated since the previous **BACKUP ARCHIVELOG ALL DELETE INPUT** command.

PLUS ARCHIVELOG is the default when backing up to non-disk device.

The process works as follows:

1. Backup all archive logs that have not yet been backed up.
2. Backup the file(s) specified in the backup command.
3. Perform a log switch.
4. Backup any remaining logs.

Backupset Backup

Change the location of a backup set

- The source for this command must be DISK
- The destination can be either DISK or SBT

```
RMAN> BACKUP ... BACKUPSET  
      {key|CompletedTimeSpec|ALL}
```

```
RMAN> BACKUP DEVICE TYPE SBT  
      BACKUPSET  
      CREATED BEFORE 'sysdate - 7'  
      DELETE INPUT;
```

ORACLE

4-23

Copyright © Oracle Corporation, 2001. All rights reserved.

Backupset Backup

The specified backup set(s) will be backed up. The source for this command must be disk.

```
RMAN> BACKUP DEVICE TYPE SBT BACKUPSET ALL;
```

This command should be run regularly as part of the production backup schedule. When it is complete, the backups exist on both disk and tape. This allows disk space management to be performed.

```
RMAN> BACKUP DEVICE TYPE SBT BACKUPSET  
      CREATED BEFORE 'sysdate-7' DELETE INPUT;
```

This command is typically run when the user wants more recent backups to exist on disk and older backups to exist on tape, but does not need backups to exist on both disk and tape at the same time.

Note: Using DELETE INPUT effectively makes this a move operation.

Restore File Optimization and Restartable Restore

- With restore file optimization, RMAN examines all target file headers before beginning the restore.
- A file is not restored if it is already in the correct location and its header contains a correct state.
- This feature allows a restore operation to be restarted after any type of failure without doing unnecessary work.
- The **RESTORE FORCE** command can be used to restore a file regardless of its state.

ORACLE

4-24

Copyright © Oracle Corporation, 2001. All rights reserved.

Restore File Optimization and Restartable Restore

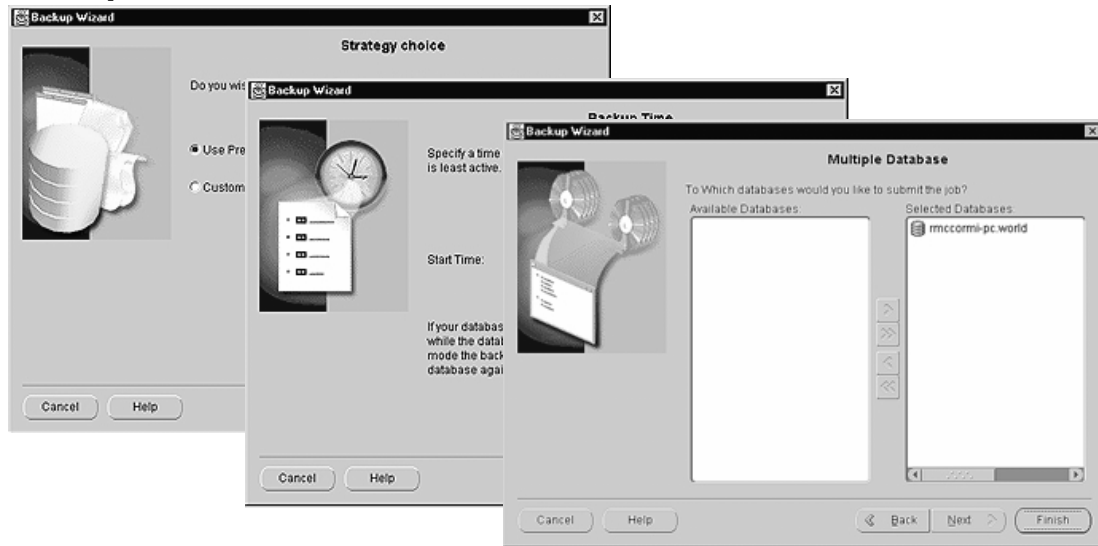
In this scenario a restore operation has been running for six hours and a tape device stops working. All except one of the necessary database files has been restored to disk. The restore job fails. You have no other choice but to run the restore task again, knowing that at least another six hours will pass before the database is running.

This is not the case with RMAN's restore file optimization. If RMAN detects that the file to be restored is consistent with the database and its header contains the expected information, it is not necessary to replace the file on disk with the backup again. In this scenario, RMAN only restores the last file. Unconditional restore of database files can be achieved by using the **FORCE** option with the **RESTORE** command.

Note: Restore optimization only checks the datafile header and does not scan the datafile body for corrupted blocks.

Recovery Manager Enterprise Manager Interface

The Enterprise Manager interface to RMAN has been improved



ORACLE

4-25

Copyright © Oracle Corporation, 2001. All rights reserved.

Recovery Manager: Enterprise Manager Interface

The Enterprise Manager interface to RMAN has been upgraded giving access to the new features in the product. The slide shows some sample screens.

RMAN Reliability Enhancements

- **Archive log failover**
- **Automatic log switch**
- **Backup piece failover**
- **Block media recovery**
- **Trial recovery**

ORACLE®

Archive Log Failover and Automatic Log Switch

- **RMAN is capable of reading multiple local archive log destination directories.**
- **If a failure is encountered while reading an archive log in one location, RMAN automatically switches to another one.**
- **RMAN only returns an error when all the local locations have failed.**
- **RMAN can delete the archive logs from all disk locations after they have been successfully backed up.**
- **At the end of any BACKUP or COPY command, RMAN automatically switches and archives the current online log.**

ORACLE

4-27

Copyright © Oracle Corporation, 2001. All rights reserved.

Archive Log Failover

It is a common practice for a database administrator to multiplex archive log locations to protect against disk failure. When disk space is no longer available, the archived logs are backed up to another storage medium and deleted from disk.

When a BACKUP ARCHIVELOG operation is invoked, it reads from one of the archive log destination directories. If RMAN finds an invalid or corrupt archive log, it uses the next multiplexed location for the backup. Only after all archive log destinations have been examined, RMAN errors out to indicate the problem with the file.

Improved archive log space management is achieved by deleting archive logs from all multiplexed destinations after a successful archive log backup. This ensures that RMAN has validated archive logs and the database administrator can feel confident that necessary recovery is possible. This behavior is triggered by the DELETE ALL INPUT option.

Also, to add further protection, at the end of any BACKUP or COPY command, RMAN automatically switches out of and archives the current online log. This automatic log switching feature improves the chances of a successful recovery.

Backup Piece Failover

- **RMAN always validates that a backup piece is not corrupt when moving it from disk to another location.**
- **If RMAN finds an invalid backup piece, it searches for a multiplexed disk copy of it.**
- **RMAN only signals an error when all of the original locations failed.**

ORACLE

4-28

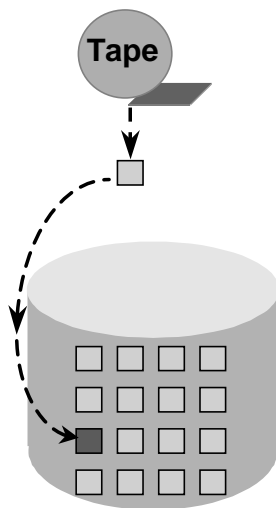
Copyright © Oracle Corporation, 2001. All rights reserved.

Backup Piece Failover

When backing up to disk, a database administrator can make multiple copies of backup pieces to different disk locations. When the primary disk space is no longer available, the on disk backups need to be deleted or copied to another storage medium. When RMAN is invoked to move backups to tape or other disk locations, it validates that the backup piece is not corrupt. If it finds an invalid backup piece, it examines the next multiplexed disk location for a valid data file backup. If none is found, RMAN errors out to indicate the problem with the file.

This ensures that RMAN has validated backups and the DBA can feel confident that necessary recovery is possible.

Block Media Recovery (BMR)



- **A block becomes the smallest unit of media restore and recovery.**
- **BMR main benefits:**
 - **Lowers the mean time to recover**
 - **Increases data availability during media recovery**
- **RMAN must be used for BMR:**
 - **Restores individual data blocks from available backups**
 - **Coordinates with the server to have them recovered**

ORACLE

4-29

Copyright © Oracle Corporation, 2001. All rights reserved.

Block Media Recovery (BMR)

BMR reduces the smallest recoverable unit of media recovery from a data file to a block. The basic premise is that when a small subset of blocks in the database are known to require media recovery, it is more efficient to selectively restore and recover just those blocks. Only blocks being recovered need to be unavailable, allowing continuous availability of the rest of the database during recovery. Block media recovery (BMR) provides two main benefits over file-level recovery:

- Lowering the mean time to recover (MTTR)
- Allowing increased availability of data during media recovery as the data file being recovered remains online.

BMR uses existing recovery mechanisms to apply changes from the redo stream to block versions restored from suitable backups. Recovery Manager (RMAN) must be used to perform BMR. The existing SQL interface for media recovery does not support BMR. RMAN restores individual data blocks from available backups and coordinates with the Oracle server process to have them recovered. If a backupset repository is maintained, RMAN can also assist in selecting the correct backup of a block to use as the starting point for media recovery.

Note: Only complete recovery is possible. Incomplete recovery would render the file physically inconsistent.

MTTR Reduction

- **Mean time to recover (MTTR) is reduced when only a small subset of blocks in a data file need media recovery.**
 - Only these blocks have to be restored from backup.
 - Redo application time is reduced by applying recovery only to the required (small) set of corrupt blocks.
- **BMR is not intended to be used when the extent of data lost or corrupted is unclear and potentially the entire file may need recovery.**
- **BMR is targeted at data losses that affect specific blocks reported in Oracle errors.**

ORACLE

4-30

Copyright © Oracle Corporation, 2001. All rights reserved.

MTTR Reduction

The primary functional requirement for BMR is to reduce MTTR in situations where only a small subset of blocks in a datafile need media recovery. Without block-level recovery, if even a single block is corrupt the administrator must restore a backup of the entire file and apply all redo changes generated for that file since backup. The reduction in MTTR realized by using block-level recovery includes both restore and recovery time:

- Only blocks needing recovery have to be restored from backup. This is only a limited performance benefit for offline backups that need to be restored from sequential access storage such as magnetic tape. With online (disk) backups, it should be significantly faster to restore a few blocks instead of the entire file.
- Only the required (small) set of corrupt blocks undergo recovery. This reduces redo application time and avoids the paging out and reading back of recovery buffers during the recovery. As with file or database media recovery, the entire redo stream still has to be read, but the reductions in redo application and recovery buffer I/O are expected to significantly lower overall recovery time as compared to file-level media recovery.

All specified blocks are recovered by a single pass through the redo stream. Currently, the user is required to enter individual block addresses in the RMAN commands for BMR.

Other BMR Benefits

- **Increased availability during media recovery:**
 - The data file where the blocks are being recovered remains online.
 - Only blocks undergoing media recovery are inaccessible.
- **Ability to skip over missing or corrupt redo:**
 - Corrupted redo records are ignored if they do not pertain to blocks being recovered.
 - BMR defers signaling a stuck recovery failure because missing redo changes may pertain to a block that could be renewed at some later point in the redo stream.

ORACLE

Other BMR Benefits

Data blocks undergoing media recovery are inaccessible to queries or DML because they are media corrupt, but the datafile itself remains online. This is a significant availability improvement over file-level recovery, where the entire datafile is offline for the duration of the recovery.

BMR may be able to survive gaps in the redo stream, if it is evident that the missing redo records do not pertain to blocks being recovered. This does not help if the redo stream is unreadable, but provides some degree of fault tolerance to missing or corrupt redo records. BMR requires an unbroken set of redo changes only for the blocks being recovered, which is a weaker requirement than the unbroken redo stream currently required by data file and database media recovery. Because failure status for each block in the set being recovered is independent, BMR may be successful for a subset of the recovery set.

BMR can defer signaling a stuck recovery failure when missing redo changes are first detected. When a block is renewed, all previous redo for that block becomes irrelevant, because the redo applies to a defunct incarnation of the block. BMR can optimistically defer failure if a redo change needed for a block is missing or corrupt, because the block can be renewed at some later point in the redo stream. For example, the Oracle server can renew a block when users delete all the rows recorded in the block or drop the table.

Recovery Manager Interface

- **RMAN supports BMR via the BLOCKRECOVER command.**
- **Identifies the backups from which to obtain the blocks to recover.**
- **Reads the backups and accumulates requested blocks into in-memory buffers.**
- **Starts and manages the block media recovery session, reading the archive logs from backup if necessary.**
- **Always does a complete recovery**

```
BLOCKRECOVER DATAFILE 7 BLOCK 3;
```

ORACLE

Recovery Manager Interface

RMAN supports BMR via the new BLOCKRECOVER command. When the user encounters a block corruption, the error message or the trace files indicate which block is causing problems. The DBA can then invoke this command to restore only the block in question, saving an enormous amount of down time and data unavailability.

The BLOCKRECOVER command does the following:

- Identifies the backups from which to obtain the blocks to recover. If the user has never used RMAN before with this database, and the only existing backups are image copies taken with Oracle7 methods, the user should use the CATALOG DATAFILECOPY command to identify those files to RMAN prior to using the BLOCKRECOVER command. This is preferable to entering the names on the BLOCKRECOVER command itself, because they only have to be entered once, and then they are automatically used for all subsequent BLOCKRECOVER commands. The CATALOG ARCHIVELOG command may also be required to specify restored archive logs.
- Reads the backups and accumulates requested blocks into in-memory buffers. If any of the desired blocks is corrupt (either media or logical corruption), RMAN reads the next oldest backup of that file, looking for a good copy of the block. The TIME option limits selection to backup sets or file copies taken at or before the specified time, SCN, or log sequence. This forces BLOCKRECOVER to use an older backup instead of the most recent one.

Recovery Manager Interface (continued)

- Starts and manages the block media recovery session, reading archive logs from backup if necessary.
- Always does a complete recovery. No point in time recovery is possible using the BLOCKRECOVER command.

Block media recovery is most useful for data losses that affect specific blocks. Typically, this type of block corruption is reported in these locations:

- Oracle error messages in standard output
- The alert log
- User trace files
- Results of the SQL commands `ANALYZE TABLE` and `ANALYZE INDEX`
- Results of the `dbverify` utility
- Third-party media management output

Block-level data loss usually results from:

- Intermittent, random I/O errors that do not cause widespread data loss
- Memory corruptions that get written to disk

For example, you may discover the following messages in a user trace file:

```
ORA-01578: ORACLE data block corrupted (file# 7, block# 3)
ORA-01110: data file 7: '/oracle/dbs/tbs_71.f'
ORA-01578: ORACLE data block corrupted (file# 2, block# 235)
ORA-01110: data file 2: '/oracle/dbs/tbs_21.f'
```

You can then specify the corrupt blocks in the BLOCKRECOVER command as follows:

```
BLOCKRECOVER DATAFILE 7 BLOCK 3 DATAFILE 2 BLOCK 235;
```

Recovery Manager Interface

- **RMAN lists blocks that failed logical validation during backup in:**
 - V\$BACKUP_CORRUPTION
 - V\$COPY_CORRUPTION
- **The CORRUPTION LIST clause specifies that all blocks listed in these views should be recovered:**

```
RMAN> BLOCKRECOVER CORRUPTION LIST  
        RESTORE UNTIL TIME 'SYSDATE - 10';
```

ORACLE

4-34

Copyright © Oracle Corporation, 2001. All rights reserved.

Recovery Manager Interface (continued)

Two types of corruption result in rows being added to V\$BACKUP_CORRUPTION and V\$COPY_CORRUPTION, by the BACKUP and COPY commands, respectively:

- Physical corruption (sometimes called media corruption). The Oracle server does not recognize the block at all: the checksum is invalid, the block contains all zeros, or the header and footer of the block do not match. Physical corruption checking is ON by default, and can be turned off with the NOCHECKSUM option.
- Logical corruption. The block has a valid checksum, the header and footer match, and so forth, but the contents are logically inconsistent. Logical checking is OFF by default, and can be turned on with the CHECK LOGICAL option.

The UNTIL clause specifies that only backups and copies created before the specified time, SCN, or log sequence number should be restored and used for the recovery.

Recovering a Group of Corrupt Blocks

This example recovers corrupt blocks in three datafiles:

```
BLOCKRECOVER DATAFILE 2 BLOCK 12, 13 DATAFILE 7 BLOCK 5, 98, 99  
DATAFILE 9 BLOCK 19;
```

Note: Each block is recovered independently during block media recovery, so recovery may be successful for a subset of blocks.

Note: RMAN cannot detect all types of corruptions.

Recovery Manager Interface (continued)

Limiting Block Media Recovery by Type of Restore

The following example recovers a series of blocks and restores only from datafile copies:

```
RUN {  
    BLOCKRECOVER  
        DATAFILE 3 BLOCK 1,2,3,4,5  
        TABLESPACE sales DBA 4194405, 4194409, 4194412  
    FROM DATAFILECOPY;  
}
```

Limiting Block Media Recovery by Backup Tag

This example recovers a series of blocks and restores only from the backup set with the tag weekly_backup:

```
BLOCKRECOVER TABLESPACE SYSTEM DBA 4194404, 4194405 FROM TAG  
"weekly_backup" ;
```

Limiting Block Media Recovery by Time, SCN or Sequence

The following example recovers two blocks in the SYSTEM tablespace and forces the blocks to be restored from backups created at least two days ago:

```
BLOCKRECOVER TABLESPACE SYSTEM DBA 4194404, 4194405 RESTORE  
UNTIL TIME 'SYSDATE-2' ;
```

The following example recovers two blocks and forces the blocks to be restored using backups made before SCN 100:

```
BLOCKRECOVER DATAFILE 9 BLOCK 13 DATAFILE 2 BLOCK 19 RESTORE  
UNTIL SCN 100 ;
```

The following example recovers two blocks and forces the blocks to be restored using backups made before log sequence 7024:

```
BLOCKRECOVER DATAFILE 9 BLOCK 13 DATAFILE 2 BLOCK 19 RESTORE  
UNTIL SEQUENCE 7024 ;
```

Trial Recovery

- Several problems may prevent media recovery from completing successfully.
- In Oracle9i, the following enhancements are implemented:
 - DBAs can invoke a Trial Recovery.
 - DBAs can instruct media recovery to mark a data block corrupt if required to proceed.
 - Recovery always leaves behind a database which can be opened.
- These enhancements are based on an optimistic redo application algorithm.
- Trial recovery does not require RMAN.

ORACLE

4-36

Copyright © Oracle Corporation, 2001. All rights reserved.

Trial Recovery

Some problems that may occur during media recovery are not *recoverable*. For example, if a redo log is somehow corrupted and recovery cannot pass one change vector in the redo stream, the recovered database becomes useless. For media recovery, the user must restore the backup and recover the database to a SCN before where the corruption occurred. For a large database, restoring a backup and recovering the database can take a long time.

The following enhancements are provided:

- If media recovery of a full database encounters a problem, recovery always leaves behind a consistent database, which can be opened read-only or with resetlogs.
- The DBA can instruct the media recovery process to mark a data block as being corrupted. The block's inconsistency is ignored and the recovery can proceed. The block will subsequently be inaccessible.
- The DBA can invoke a Trial Recovery in order to investigate if the recovery problem is an isolated problem.

The optimistic redo application algorithm assumes that no problem will occur during media recovery. If any problem does occur, the Oracle server will automatically undo the last applied changes, and not leave an inconsistency in the recovered database.

Trial Recovery

Allowing Corruptions of Data Blocks

- If recovery encounters a problem, the database is, first of all, left in a physically consistent state.
- The alert log indicates if it is possible to bypass the problem by marking a data block as corrupt.
- The alert log also identifies the data block.
- You can open the database in read-only mode before deciding whether to open resetlogs or to corrupt the block.

```
SQL> RECOVER DATABASE ALLOW 1 CORRUPTION;
```

ORACLE

4-37

Copyright © Oracle Corporation, 2001. All rights reserved.

Trial Recovery: Allowing Corruptions of Data Blocks

If media recovery encounters a problem, either a stuck recovery problem or a problem during redo application, recovery always stops and leaves behind a consistent database.

The database provides a lot of information about the problem in the alert logs. The alert log indicates if recovery is capable of recovering past the problem by marking as corrupted the data block causing the problem. The alert log also provides information about the block: its file name, file number, block number, block type, data object number, and so on.

Users can then investigate the impact of corrupting the problematic block according to the information provided in the alert logs. Users can also open the database read-only (provided they have recovered all the database files to the same point in time) and query the database to see to what table this data block belongs.

However, the alert log only has information about the errors encountered so far. A better approach is to use Trial Recovery so you can assess the entire extent of damage.

After the investigation, users can either choose to open resetlogs, or continue recovery with a new recovery option `ALLOW N CORRUPTION`. For example, the following command marks up to one data block corrupt and proceeds with recovery:

```
SQL> RECOVER DATABASE ALLOW 1 CORRUPTION ;
```

Trial Recovery

- If only a few blocks are bad, corrupting them and recovering the rest is a good solution.
- If there are many corrupted blocks, a new restore or open resetlogs may be a better option.
- Trial recovery is a new concept that allows the user to find out how many more problematic blocks are going to pop-up during recovery.
- Trial recovery only changes blocks in memory. Updated blocks are not written to disk.
- Encountered errors are reported in the alert log.

```
RECOVER DATABASE ... TEST;
```

ORACLE

4-38

Copyright © Oracle Corporation, 2001. All rights reserved.

Trial Recovery

When problems such as stuck recovery occur during media recovery, the user is confronted with a difficult choice. If the block is not that important, and this is an isolated problem, it is better to mark the block corrupt. But if this is not an isolated problem or the block is an important one in the SYSTEM tablespace, it may be better to restore another backup of the corrupted files or to open resetlogs (this solution discards all changes after the problem redo that generated the error, but guarantees a logically consistent database). Until now, it was difficult for a user to tell whether a stuck recovery was an isolated problem.

To address this problem, the concept of Trial Recovery is introduced. The Trial Recovery applies redo in a way similar to normal media recovery, but it never writes its changes to disk, and it always rolls back its changes at the end of the test. Trial Recovery is designed to allow a DBA to peek ahead in the redo stream to see if there are additional problems.

Users may invoke Trial Recovery by adding the TEST option to any restore command:

```
RECOVER DATABASE USING BACKUP CONTROLFILE TEST;  
RECOVER TABLESPACE sales TEST;  
RECOVER DATABASE UNTIL CANCEL TEST;
```

Note: You can run the Trial Recovery command at any time that you can run a normal recovery command. Nevertheless, you should only need to run test recovery when recovery runs into problems.

Trial Recovery (continued)

By default, a Trial Recovery always attempts to corrupt blocks in memory if this action allows the RECOVER ... TEST statement to proceed. In other words, Trial Recovery by default can corrupt an unlimited number of data blocks. You can specify the ALLOW n CORRUPTION clause on the RECOVER ... TEST statement to limit the number of data blocks Trial Recovery can corrupt in memory.

Errors during the Trial Recovery are written to alert files, which are clearly marked as test run errors. The same as in a normal recovery, a Trial Recovery may prompt users for logs and accept file names from the user. A Trial Recovery may end when either :

- The Oracle server runs out of available buffers in memory
- An unrecoverable error is signaled
- The user cancels or interrupts the recovery session
- The next redo record in the redo stream changes the control file
- All redo desired to be applied has been applied

When a Trial Recovery ends, except the possible error messages in alert files, all effects of the test run are automatically removed from the system. If the instance dies, all effects of the Trial Recovery are also removed from the system because a Trial Recovery never writes changes to disk.

Miscellaneous RMAN Enhancements

- **New reporting capabilities:**
 - `REPORT OBSOLETE`
 - `REPORT NEED BACKUP`
- **RMAN catalog maintenance:**
 - **Command unification**
 - **LIST command improvements**
 - **New SHOW command**
 - **CROSSCHECK autolocate**

ORACLE

4-40

Copyright © Oracle Corporation, 2001. All rights reserved.

Sundry RMAN Enhancements

Two reporting commands have been enhanced:

- `REPORT OBSOLETE`
- `REPORT NEED BACKUP`

The `LIST`, `CHANGE`, `CROSSCHECK`, and `DELETE` commands were not consistent in the types of backup objects they could process, the syntax that was used to process various objects, and the status codes that could be assigned to different objects.

All types of objects that can be created by RMAN (datafile copies, archived logs, backup pieces, and proxy copies), can now be processed with consistent syntax, and all status codes (available, unavailable, and expired) can apply to all types of objects.

REPORT OBSOLETE

- This command shows which backups are no longer needed according to the retention policy:

```
REPORT OBSOLETE  
    <REDUNDANCY n|RECOVERY WINDOW n>;
```

- New option RECOVERY WINDOW
- Reported files include datafiles, archive logs, or file copies
- New DELETE OBSOLETE command

ORACLE

4-41

Copyright © Oracle Corporation, 2001. All rights reserved.

REPORT OBSOLETE

REPORT OBSOLETE is the RMAN command that shows which backups are no longer needed.

Prior to Oracle9i, RMAN already had a REPORT OBSOLETE command that could be used to implement retention policies. However this command had two main drawbacks:

- It reported data file backups and copies but not archive logs and archive log backups.
- It printed a list of obsolete backups but provided no way to delete them.

Both of these drawbacks have been overcome.

First, a new option, RECOVERY WINDOW, has been added to specify a window of time during which the database must be recoverable. This option is mutually exclusive with the REDUNDANCY option. When RECOVERY WINDOW is specified, one backup that is older than the recovery window must exist for each datafile. All backups older than that one are obsolete.

In addition to obsolete datafile backups, RMAN displays obsolete archive logs and archive log backups. Regardless of which retention policy is used (redundancy or recovery window), RMAN uses that policy to determine which backups and copies of datafiles are no longer needed, which in turn determines when archive logs and their backups are no longer needed. The creation of a datafile backup is considered when deciding which logs to keep.

REPORT OBSOLETE (continued)

Example

If none of UNTIL, REDUNDANCY, or RECOVERY WINDOW is specified retention policy specified by CONFIGURE RETENTION POLICY is used. If the user has set the retention policy to NONE, then RMAN signals an error.

The UNTIL and REDUNDANCY options are still supported for backwards compatibility.

Finally, a new DELETE OBSOLETE command is provided to delete files that would be reported by REPORT OBSOLETE.

The following example reports obsolete backups and copies with a redundancy of 1:

```
RMAN> REPORT OBSOLETE;
```

Report of obsolete backups and copies

Type	Key	Completion Time	Filename/Handle
-----	-----	-----	-----
Backup Set	836	04-OCT-00	
Backup Piece	839	04-OCT-00	/home/oracle/dbs/05aetj6b_1_1
Backup Set	807	04-OCT-00	
Backup Piece	810	04-OCT-00	/home/oracle/dbs/03aetj1f_1_1

REPORT NEED BACKUP

- Reports files that need backup according to the retention policy
- If the retention policy is set to NONE, an error is signaled.
- Example: report files that need to be backed up to satisfy a retention policy of 5 days:

```
REPORT NEED BACKUP RECOVERY WINDOW OF 5 DAYS ;
```

ORACLE

4-43

Copyright © Oracle Corporation, 2001. All rights reserved.

REPORT NEED BACKUP

If the retention policy is set to some value (not NONE) then you can simply type REPORT NEED BACKUP and RMAN takes into account the currently configured retention policy. (If the retention policy is set to none, you will get an error.)

This example reports all datafiles in the database that require the application of five or more incremental backups to be recovered to their current state:

```
RMAN> REPORT NEED BACKUP INCREMENTAL 5 DATABASE;
```

Report of files that need more than 5 incrementals during recovery

```
File Incrementals Name
```

```
-----  
1      9      /home/oracle/dbs/tbs_01.f  
2      9      /home/oracle/dbs/tbs_02.f  
3      9      /home/oracle/dbs/tbs_11.f
```

Command Unification

- **RMAN has four related command sets for catalog maintenance: LIST, CROSSCHECK, DELETE EXPIRED, and CHANGE.**
- **Changes have been implemented to provide greater consistency:**
 - Their actions have been unified.
 - They use similar syntax.
 - Output formats and status codes are shared when possible.
- **The old syntax can still be used for backwards compatibility.**

ORACLE

4-44

Copyright © Oracle Corporation, 2001. All rights reserved.

Command Unification

The following changes have been implemented to provide greater consolidation among the command sets:

- CROSSCHECK and DELETE EXPIRED deal with all object types.
- CROSSCHECK, DELETE, and LIST are able to process smaller object sets like CHANGE does.
- CROSSCHECK sets status to 'X' if a datafile copy, controlfile copy, and so on, is not available and skips any objects that are unavailable.
- CROSSCHECK and DELETE functions have been consolidated. UNCATALOG function has been removed.
- DELETE lists and prompts for confirmation unless NOPROMPT is specified.
- CHANGE syntax enhanced to be more consistent with DELETE/CROSSCHECK and to support a larger object list.
- LIKE (fileNamePattern) added to CROSSCHECK/DELETE/CHANGE
- LIST shows what CROSSCHECK/DELETE EXPIRED will process, and so on.

LIST Command Improvements

- **BY BACKUP output orientation shows:**
 - Backup sets and the contents of each backup set (pieces and files)
 - Backup copies of any file
- **BY FILE output orientation shows:**
 - The file name
 - Backup sets where the file appears
 - Backup copies of this file
- **A SUMMARY option is available with the BY BACKUP orientation, which gives a one-line summary for each file or backup set.**

ORACLE

LIST Command Improvements

The user can now choose between **BY BACKUP** and **BY FILE** output orientation.

BY BACKUP is the default as long as there is no **OF** option specified in the **LIST** command, otherwise **BY FILE** is the default.

LIST Command New Syntax

```
LIST BACKUP .. [listoptions];  
  
listoptions: [BY <BACKUP|FILE>]  
            [SUMMARY | VERBOSE]
```

- **Note that BY and SUMMARY | VERBOSE options only apply to LIST BACKUP, not to LIST COPY.**
- **SUMMARY | VERBOSE only applies to BY BACKUP orientation.**
- **Defaults: BY BACKUP, VERBOSE**

ORACLE

LIST Command Examples

The following example displays the backup information oriented by backup. It displays four different backup sets and the files contained in each one.

```
RMAN> LIST BACKUP;
```

List of Backup Sets

=====

BS Key	Device	Type	Elapsed Time	Completion Time
--------	--------	------	--------------	-----------------

387	SBT_TAPE		00:00:03	15-SEP-00
-----	----------	--	----------	-----------

BP Key: 388	Status: AVAILABLE	Tag:
-------------	-------------------	------

Piece Name: 12c5erb2_1_1

List of Archived Logs in backup set 387

Thrd	Seq	Low SCN	Low Time	Next SCN	Next Time
------	-----	---------	----------	----------	-----------

1	144	51234	15-SEP-00	51318	15-SEP-00
---	-----	-------	-----------	-------	-----------

1	145	51318	15-SEP-00	51324	15-SEP-00
---	-----	-------	-----------	-------	-----------

1	146	51324	15-SEP-00	51330	15-SEP-00
---	-----	-------	-----------	-------	-----------

BS Key	Type	LV	Size	Device	Type	Elapsed Time	Comple. Time
--------	------	----	------	--------	------	--------------	--------------

396	Full		60M	SBT_TAPE		00:00:19	15-SEP-00
-----	------	--	-----	----------	--	----------	-----------

BP Key: 397	Status: AVAILABLE	Tag:
-------------	-------------------	------

Piece Name: 13c5erba_1_1

List of Datafiles in backup set 396

File	LV	Type	Ckp SCN	Ckp Time	Name
------	----	------	---------	----------	------

1	0	Full	51333	15-SEP-00	/oracle/dbs/tbs_01.f
---	---	------	-------	-----------	----------------------

2	0	Full	51333	15-SEP-00	/oracle/dbs/tbs_02.f
---	---	------	-------	-----------	----------------------

3	0	Full	51333	15-SEP-00	/oracle/dbs/tbs_11.f
---	---	------	-------	-----------	----------------------

4	0	Full	51333	15-SEP-00	/oracle/dbs/tbs_12.f
---	---	------	-------	-----------	----------------------

5	0	Full	51333	15-SEP-00	/oracle/dbs/tbs_21.f
---	---	------	-------	-----------	----------------------

BS Key	Device	Type	Elapsed Time	Completion Time
--------	--------	------	--------------	-----------------

423	SBT_TAPE		00:00:01	15-SEP-00
-----	----------	--	----------	-----------

BP Key: 424	Status: AVAILABLE	Tag:
-------------	-------------------	------

Piece Name: 14c5erce_1_1

List of Archived Logs in backup set 423

Thrd	Seq	Low SCN	Low Time	Next SCN	Next Time
------	-----	---------	----------	----------	-----------

1	147	51330	15-SEP-00	51336	15-SEP-00
---	-----	-------	-----------	-------	-----------

LIST Command Examples (continued)

```

BS Key   Type LV Size          Device Type Elapsed Time Complet. Time
-----
427      Full  1M           SBT_TAPE    00:00:02      15-SEP-00
        BP Key: 428   Status: AVAILABLE   Tag:
        Piece Name: c-674966176-20000915-00

```

Controlfile Included: Ckp SCN: 51338 Ckp time: 15-SEP-00

The following example displays the backup information oriented by file. It groups the files in three different sections depending on the file type.

RMAN> LIST BACKUP BY FILE;

List of Datafile Backups

=====

File Key	TY	LV	S	Ckp SCN	Ckp Time	#Pieces	#Copies	Tag
1	797	B	1	A 204376	06/28/2000 16:56:25	1	1	
	796	B	1	A 204373	06/28/2000 16:55:28	1	2	
	788	B	F	A 41469	06/18/2000 01:51:28	1	1	
	208	P	F	A 61506	06/22/2000 22:52:25			

List of Archived Log Backups

=====

Thrd	Seq	Low SCN	Low Time	BS Key	S	#Pieces	#Copies	Tag
1	125	39578	06/17/2000 09:49:09	791	A	1	7	
				790	A	1	1	
1	126	40097	06/17/2000 10:49:09	791	A	1	7	
				790	A	1	1	

List of Controlfile Backups

=====

CF	Ckp SCN	Ckp Time	BS Key	S	#Pieces	#Copies	Tag
204375	06/28/2000 16:56:25	797	A	1	1		
204372	06/28/2000 16:55:28	796	A	1	2		
61508	06/23/2000 00:53:50	795	A	1	1		

Note: Refer to the Chapter 17 of the *Oracle9i Recovery Manager User's Guide and Reference* for an explanation of the various column headings in the LIST output.

New SHOW Command

Displays current values for various CONFIGURE commands:

```
SHOW show_operand [ ,show_operand ...];
```

```
show_operand: RETENTION POLICY      |
               EXCLUDE               |
               BACKUP COPIES         |
               CHANNEL               |
               DEFAULT DEVICE TYPE   |
               DEVICE TYPE           |
               SNAPSHOT CONTROLFILE  |
               ...                   |
               ALL
```

ORACLE

New SHOW Command

The operands provide the following information:

- | | |
|--|---|
| • RETENTION POLICY: | Retention policy for the current target database |
| • EXCLUDE: | Tablespaces which are excluded from whole database backups |
| • DATAFILE ARCHIVELOG
that BACKUP COPIES: | Number of identical copies of each backup will be created |
| • DEVICE TYPE: | Device type that has been configured |
| • DEFAULT DEVICE TYPE: | Device type that will be used for automated backups |
| • CHANNEL: | Channel(s) that will be used for automated backup/restore |
| • SNAPSHOT CONTROLFILE: | Name of the snapshot controlfile |
| • AUXNAME: | Names of any files that have been configured for the auxiliary instance |
| • MAXSETSIZE: | Maximum backup set size |
| • BACKUP OPTIMIZATION: | Whether backup optimization is turned on |
| • ALL: | All of the above information |

SHOW Command Example (continued)

```
RMAN> SHOW ALL;
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT';
CONFIGURE DEVICE TYPE 'SBT' PARALLELISM 1;
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;
CONFIGURE DATAFILE BACKUP COPIES FOR DISK TO 2;
CONFIGURE DATAFILE BACKUP COPIES FOR SBT TO 1; #default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR SBT TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DISK TO 1; # default
CONFIGURE MAXSETSIZE TO 3072K;
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/oracle/dbs/cf_snap.f';
CONFIGURE EXCLUDE FOR TABLESPACE 'TBS_5';
```

Note: The output is displayed so that you can paste it into a script and run it as an RMAN command file. You can even run the script on a different target database.

This shows only defaults, no configure commands have been run:

```
RMAN> show all;
RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK
TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1;
# default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO
1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/databases/oracle9i/dbs/snapcf_ed27.f'; # default
```


CROSSCHECK Autolocate

- **Finds backup pieces in various Real Application Clusters nodes or devices automatically.**
- **Channels are automatically allocated if configured.**
- **RELEASE CHANNEL deallocates all channels.**
- **CROSSCHECK commands check all the channels for each job.**
- **The first channel to find the backup piece is where the operation is performed.**

ORACLE

CROSSCHECK Autolocate

Sometimes backups and copies disappear from disk or tapes in the media management library become unavailable. For example, someone may inadvertently delete backup pieces from disk, or one of the tapes used by the media manager may become corrupted. To ensure that data about backup sets and image copies in the recovery catalog or control file are synchronized with actual files on disk or in the media management catalog, you can use the crosscheck command.

In a Real Application Clusters environment, an object might exist on only one of the nodes. Also, backups might exist on both disk and tape. Therefore, RMAN needs to allow multiple maintenance channels, and DELETE and CROSSCHECK need to check all the channels for each object. The first channel to find it is where the operation is performed.

If the object is not found on any channel, the operation is attempted on the last channel of the appropriate device type to check the object. If the command is DELETE, the operation fails; otherwise CROSSCHECK marks the object EXPIRED. If the command is DELETE EXPIRED and the object exists, the operation fails. The rationale for the delete behavior is that the RMAN recovery catalog is in synchronization (or control file, if no recovery catalog), and therefore it is possible it really exists, but the proper channel was not allocated with the correct parameters. And if it is not in synchronization, CROSSCHECK should be used to force it into synchronization first.

CROSSCHECK Examples

In Oracle release 8.1.7, this enhancement is done for backup sets, backup pieces, and proxy copies. In Oracle9i, this was expanded to include data file copies, control file copies, and archived logs.

This is useful when the media manager deletes backups without letting RMAN know.

The following example queries the status of all backups on disk on two different nodes:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 2;  
CONFIGURE CHANNEL 1 DEVICE TYPE DISK CONNECT  
'sys/sys_pwd@node_1';  
CONFIGURE CHANNEL 2 DEVICE TYPE DISK CONNECT  
'sys/sys_pwd@node_2';  
CROSSCHECK BACKUP;  
RELEASE CHANNEL;
```

Other RMAN Enhancements

- **Messages and debugging have been improved.**
- **Support for Oracle Managed Files (OMF)**
- **Multiple block size support**
- **Default mode is NOCATALOG**

ORACLE

4-53

Copyright © Oracle Corporation, 2001. All rights reserved.

Other RMAN Enhancements

Messages and Debugging

RMAN no longer prints the RMAN-`nnnnn` prefix for nonerror messages, and certain messages that conveyed no useful information to the user are no longer shown.

Oracle Managed File Support

RMAN supports the new Oracle Managed Files (OMF) feature by allowing data files to be restored directly to OMF file names.

Multiple Block Size Support

RMAN supports the new multiple block size feature. If files with more than one block size are backed up in the same operation, they are automatically segregated into separate backup sets.

Default Mode Is NOCATALOG

All that is necessary to use RMAN is to start the RMAN executable and connect to the target database. If neither CATALOG or NOCATALOG is specified, then RMAN automatically begins using the control file as the backup repository as soon as the first command requiring a repository is entered.

Summary

In this lesson, you should have learned how to:

- **Explain automatic channel allocation**
- **Define and configure a retention policy**
- **Describe new backup features like long term backups, restartable backup and restore, and so on.**
- **Perform Block Media Recovery using RMAN**
- **Administer the catalog using the new options of list, delete, and change commands**
- **Perform a Trial Recovery**

ORACLE

Practice 4-1 Overview

This practice covers the following topics:

- **Checking the RMAN configuration**
- **Doing a backup without a Recovery Catalog**
- **Doing a restore without a Recovery Catalog**
- **Doing a test recovery**
- **Doing a complete recovery**

ORACLE

5

Oracle9i Data Guard

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Understand the Oracle9i Data Guard Architecture**
- **Configure the physical Standby Database in no-data-loss mode**
- **Initiate a Database Switchover operation**
- **Set up automatic archive gaps detection**

ORACLE

Objectives

- **Launch managed recovery mode in the background**
- **Apply a delay to redo application on the standby site**
- **Create or register standby redo logs**
- **Set up new initialization parameters as well as their new attributes**

ORACLE

Oracle9i Data Guard Overview

Data Guard protects your data by:

- Providing easy configuration and control with the GUI interface
- Providing switchover to another site for planned maintenance
- Providing failover to another site during unplanned failures
- Guaranteeing no data loss through a synchronous log transport
- Preventing the propagation of mistakes and corruptions



ORACLE

5-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Data Guard Overview

Oracle9i Data Guard helps your database survive possibly destructive events by:

- Enhancing high availability and disaster protection through a distributed computing configuration.

A Data Guard configuration consists of two loosely-connected systems, called sites, that combine the production database and the physical standby database into a single, easily managed disaster recovery solution.

- Providing a single, uniform management interface.

The Oracle Data Guard broker is the component that helps you to configure database resources into a single unit of failover across a Data Guard configuration and incorporate application resources into the broker's monitoring and control. The broker provides two optional interfaces: the Oracle9i Data Guard Manager graphical-user interface and the Oracle9i Data Guard DGMGRL command-line interface.

- Automating the creation and configuration of physical standby databases.

Use the broker to create the database and to manage the database and applications on each site. One site is designated a primary site, where the database is available to applications and from where the Oracle Data Guard log transport services ship the data in the form of archived redo logs. You can configure and instantiate from one to nine additional server sites to serve as standby systems to the primary site. The standby sites accept redo logs shipped from the primary site and apply changes to their copies of the database.

Oracle9i Data Guard Overview (continued)

- Providing a single unit of failover that groups dependent database and application resources as an atomic unit of failover (including failover policies that you can customize). Also, support for planned switchover helps to reduce downtime during routine maintenance operations.

The failover and switchover capabilities allow you to determine how the role of the primary database site should migrate in the event of a failure. With the broker, you can define and automate the primary role migration by implementing an n-way failover system with replicated data across all of the server nodes.

- Configuring redo log transport services and log apply services.

You can set up the standby database to use various transport modes ranging from batch data copy mode to synchronous data copy mode. This will be detailed later in this lesson

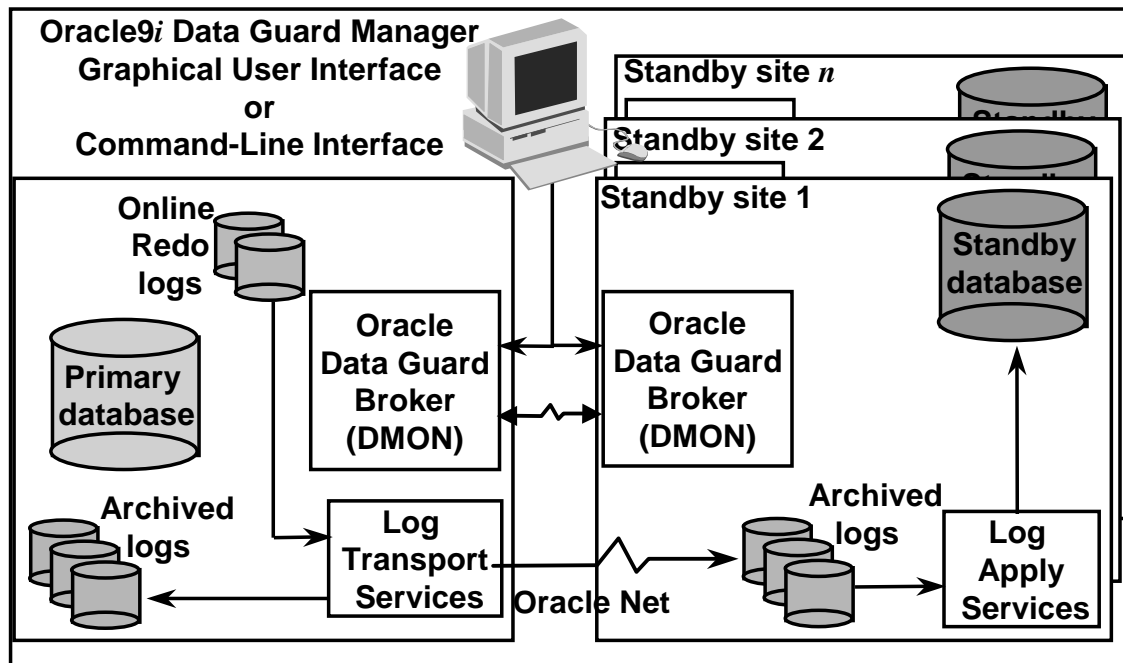
- Managing an extensible environment.

DBAs can add and remove resources (databases and applications) and sites, and configure whole site failover without any downtime occurring on the existing configuration.

- Providing monitoring, alert, and control mechanisms.

Besides automating the complex task of configuration, the broker automates the management and operational tasks a DBA must perform across the multiple sites in a Data Guard configuration. The broker monitors and controls all of the systems within a single standby database configuration.

Oracle9i Data Guard Architecture



ORACLE

5-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Data Guard Architecture

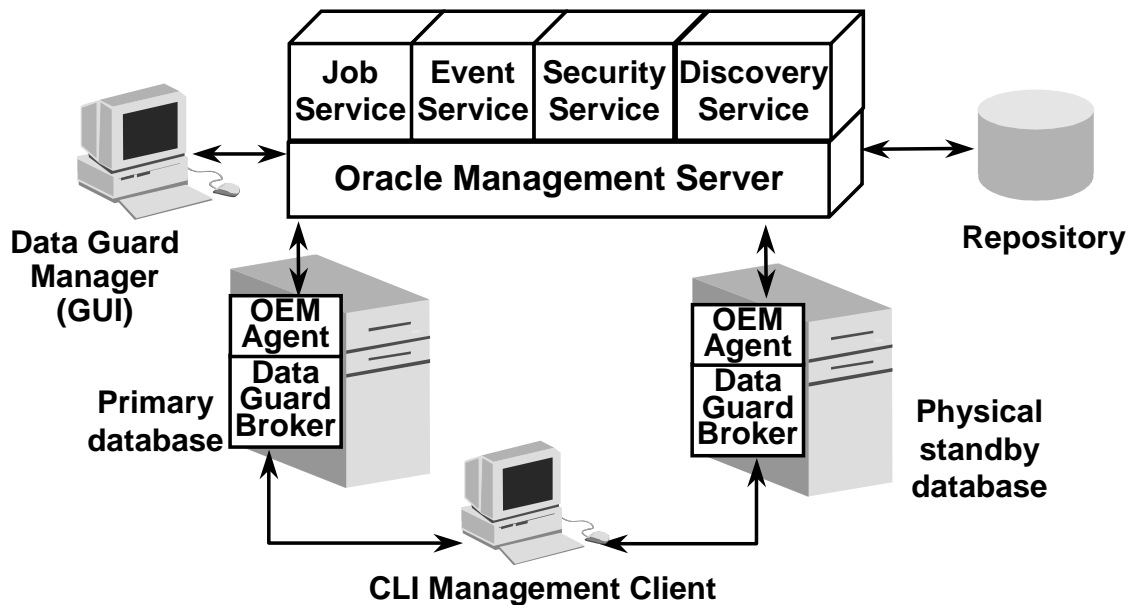
The Oracle9i Data Guard architecture consists of the following components:

- **Primary database:** A primary database is a production database. The primary database is used to create a standby database. Every standby database is associated with one and only one primary database.
- **Physical standby database:** A physical standby database is a database replica created from a backup of a primary database
- **Log transport services:** Log transport services control the automated transfer of archived redo logs from the primary database to one or more standby sites or destinations
- **Network configuration:** The primary database is connected to one or more remote standby databases through Oracle Net
- **Log apply services:** Log apply services apply the archived redo logs to the standby database
- **Data Guard broker:** Data Guard broker is the management and monitoring component that helps you configure, control, and monitor a fault tolerant system consisting of a primary database protected by one or more physical standby databases. The broker automates the previously manual process of configuration. Once it has created the Data Guard configuration, the broker monitors the activity, health, and availability for all systems in the Data Guard configuration. You can control the Data Guard Broker by using the Oracle9i Data Guard Manager tool (GUI or CLI versions).

Note: The DMON process is started using the initialization parameter `DRS_START=TRUE`

Oracle9i: New Features for Administrators 5-6

Data Guard Broker and Data Guard Manager



ORACLE

5-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Guard Broker and Data Guard Manager

This slide displays the relationship between the Oracle Management Server, Data Guard Broker, and Data Guard Manager. The broker code is part of each Oracle server process. This framework of processes is collectively known as the Data Guard Broker. A client tool (dgmgrl) can connect to each individual process, one at a time, using the command line interface.

Through Data Guard Manager, it is possible to access all databases simultaneously. It is necessary to use Oracle Management Server for Data Guard Manager (GUI).

Note: It is not necessary to use the Oracle Management Server for Data Guard command-line interface.

Oracle9i Data Guard Broker

- This is a new component which provides monitoring, control, and automation layered above the log transport services.
- You can control the entire process from the primary to either type of standby database.
- A process resides on each server and their combination forms the broker framework.
- There are two interfaces to the broker:
 - Data Guard Manager (GUI)
 - DGMGRL Command Line Interface

ORACLE

5-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Data Guard Broker

Data Guard broker provides monitoring, control, and automation layered above the log transport services and the physical standby components.

Data Guard Broker has these interfaces:

- Data Guard Manager (GUI)
 - Configuration and setup tasks
 - Operational tasks
- DGMGRL Command Line Interface (CLI)
 - Basic monitoring
 - Commands required to execute role changes and to set up the Data Guard environment

Refer to the *Oracle9i Data Guard Broker Guide* for additional information on how to use these interfaces.

Oracle9i Data Guard Broker

- **Manages the databases in the environment through a unified configuration.**
- **Enables CLI access, using any client for:**
 - **V\$ view information**
 - **Role change initiation**
 - **Broker configuration file**

ORACLE

5-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Data Guard Broker (continued)

Oracle9i Data Guard Broker manages all databases in the environment through a unified, data-protected configuration.

It enables CLI access using any client through any system in the Data Guard configuration. If one system fails, any of the others can be used for access.

- Access through the broker CLI provides V\$ view information as well as role change initiation. This information includes V\$ views that can be joined across systems to retrieve an end-to-end view of primary log send queue depth, the standby apply queue depth, as well as data specific to physical standby database.
- The broker configuration file is kept and synchronized on each system in the configuration. This file contains the desired, DBA-specific configuration for all systems within a Data Guard configuration. When a DBA specifies a new desired configuration (for example, a role change such as a failover), all configuration files are updated. As a result, if one or more systems fail, the remaining accessible systems provide the last desired state, and (from the Data Guard alert log), what has happened across all monitored systems.

Oracle9i Data Guard Manager

This is the GUI interface which provides:

- **A complete monitoring tool**
- **A setup and configuration wizard**
- **Elimination of single point of failure**
- **The Enterprise Manager Event system that enables the easy creation of events and their polling intervals**

ORACLE

5-10

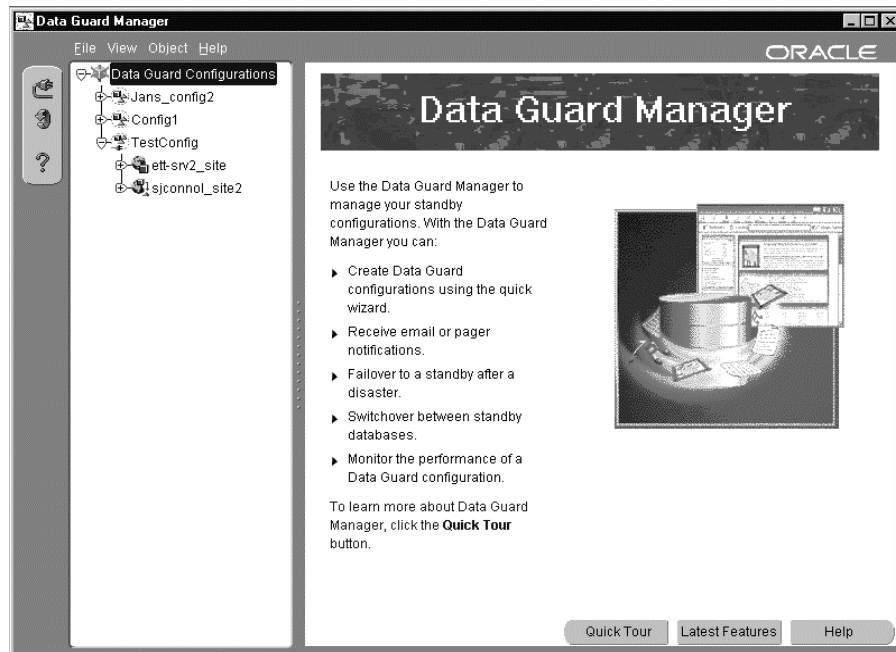
Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle9i Data Guard Manager

Oracle9i Data Guard Manager provides the following functionality:

- A complete monitoring view of the topology of the environment. It provides the only unified, one-stop view of status. All other access methods require queries across systems to produce an end-to-end view of progress.
- A wizard that guides users through multinode standby setup and configuration. This provides a dramatically faster path to implementation than the current Oracle8i Automated Standby Database process.
- Elimination of a single point of failure, except for the agent in very specific scenarios. As in other environments, the Enterprise Manager framework offers the ability to eliminate any other single point of failure by configuring the Enterprise Manager client to more than one Oracle Management Server (OMS). Like all Oracle databases, the Enterprise Manager repository that holds scheduling and status information can be configured to minimize downtime using either high availability or disaster protection solutions, or both.
- The Enterprise Manager Event system that enables the easy creation of events and their polling intervals. The Enterprise Manager Event system offers a base set of event tests that can be run on managed services to check for specific conditions, such as a database down, 24-hours a day. The intuitive interface allows users to register events on multiple services with a customizable polling interval. Depending on the event test being run, users can also set thresholds for when they want to be notified.

Oracle9i Data Guard Manager



Oracle9i Data Guard Manager (continued)

The slide shows a sample screen shot of the GUI interface to the broker services. This interface runs from Enterprise Manager and requires the Oracle Management Server.

No Data Loss and No Data Divergence Definitions

- **Standby Database can sometimes lag behind the primary database.**
- **In case of Primary destruction, data will be lost.**
- **With no-data-loss, primary database modifications can only be committed when corresponding redo information is also available on the standby database.**
- **No-data-divergence extends no-data-loss by prohibiting primary database modifications when connectivity to standby database is not available.**

ORACLE

No Data Loss and No Data Divergence Definitions

The primary and standby databases are synchronized by applying redo logs from the primary database to the standby database. Although the goal is to keep the databases identical, the transactions applied to the standby database can sometimes lag behind the primary database.

This lag may occur either because the data has not yet reached the standby site, or it may have reached the standby site, but has not yet been applied to the standby database. If the data needed to keep the databases synchronized is not yet available on the standby site, and you must fail over, the contents of the databases will diverge, and some data will be lost.

With Oracle9i Data Guard, varying degrees of potential data loss can be configured, from no data loss to minimal data loss. Each degree of potential data loss has a varying effect on primary database performance.

The definition of no data loss is deceptively simple and very subtle. Log transport services will not commit application modifications to the primary database until the modifications are also available on a standby database. No data loss does not imply that the data modifications have been applied to the standby database, but that the data is available to be applied to the standby database.

The definition of data divergence extends the concept of no data loss. Data divergence occurs when data modifications occur on the primary database when connectivity to the standby database is not available. No data divergence prohibits primary database modifications when connectivity to at least one standby database is not available. In other words, the data on the primary database is protected against both loss and data divergence.

Data Availability Modes in Data Guard

- **Guaranteed protection:**
 - No data divergence
 - LGWR send redo records to standby
- **Instant protection:**
 - No data loss
 - LGWR send redo records to standby
- **Rapid protection:**
 - LGWR slaves send redo records to standby
 - No guaranty for primary modifications to be available on standby when primary commits
- **Delayed protection: Same as in Oracle8i**

ORACLE

Data Availability Modes in Data Guard

The following data availability modes can be configured in the Data Guard environment:

- **Guaranteed protection:** The standby database cannot diverge from the primary database at all. If a standby database is unavailable, processing automatically halts on the primary database as well. When operating in this mode, the log writer process transmits redo records from the primary database to the standby database, and a transaction is not committed on the primary database until it has been confirmed that the transaction data is available on at least one standby database. This has the potential to affect primary database performance, but provides the highest degree of data availability at the standby site.
- **Instant protection:** The standby database may temporarily diverge from the primary database, but upon failover to the standby database, the databases can be synchronized, and no data will be lost. As with guaranteed protection, the log writer process transmits redo logs from the primary database to the standby database. The transaction is not committed on the primary database until it has been confirmed that the transaction data is available on at least one standby database.
- **Rapid protection:** The log writer slave process transmits redo logs to the standby sites without regard to the data availability on any standby database.
- **Delayed protection:** The archiver process transmits the completed archives to the standby sites. This lowest level of protection has been available in releases prior to Oracle9i.

Data Availability Mode Configuration Process

You need to specify a:

- Redo Log Writing Process (LGWR | ARCH)

```
LOG_ARCHIVE_DEST_n = 'SERVICE=stby1 LGWR'
```

- Network Transmission Mode (SYNC | ASYNC [=Blocks])

```
LOG_ARCHIVE_DEST_n = 'SERVICE=stby1 LGWR SYNC'
```

- Method of Writing Archive Logs to Disk ([NO]AFFIRM)

```
LOG_ARCHIVE_DEST_n = 'SERVICE=stby1 LGWR SYNC AFFIRM'
```

- Redo Log Reception Option
 - Standby redo logs or normal archived redo logs
- Failure Resolution Policy ([UN]PROTECTED)

```
ALTER DATABASE SET STANDBY DATABASE PROTECTED;
```

ORACLE

Data Availability Mode Configuration Process

In order to configure a Data Availability mode with Oracle9i Data Guard, you need to process as follow:

- Specifying a Redo Log Writing Process: The log writer process propagates primary database redo log changes to one or more standby databases. This is achieved using the LGWR attribute of the LOG_ARCHIVE_DEST_n initialization parameters as shown in the above example. ARCH attribute is the default value as in Oracle8i.
- Specifying a Network Transmission Mode: When using the log writer process to archive redo logs, the DBA can specify synchronous (SYNC) or asynchronous (ASYNC) network transmission of redo logs to archiving destinations. The SYNC attribute indicates that all network I/O operations are to be performed synchronously, in conjunction with each write operation to the online redo log. When operating in this mode, and upon a commit statement, control is not returned to the executing application or user until the redo information is received by the standby site. This attribute has the potential to affect primary database performance adversely, but provides the highest degree of data availability at the destination site. The ASYNC=blocks keyword indicates that all network I/O operations are performed asynchronously, and control is returned to the executing application or user immediately. You can specify a block count to determine the size of the SGA network buffer to be used. Block counts from 0 to 2048 are allowed. In general, for slower network connections, use larger block counts.

Data Availability Mode Configuration Process (continued)

- Specifying a Method of Writing Archive Logs to Disk: The `[NO]AFFIRM` attribute (`AFFIRM` is the default) is used to specify whether log archiving disk write I/O operations are to be performed synchronously or asynchronously. By default, disk write operations are performed asynchronously. It is necessary to receive acknowledgment of the availability of the modifications on the standby database in a no-data-loss environment. This is achieved using the `SYNC` and `AFFIRM` attributes of the `LOG_ARCHIVE_DEST_n` initialization parameters. The `SYNC` attribute indicates that synchronous network transmission is necessary, and the `AFFIRM` attribute indicates that synchronous archive redo log disk write I/O operations are necessary. Together, these attributes ensure that primary database modifications are available on the standby database.
- Specifying a Redo Log Reception Option: Redo logs transferred from the primary database are received by the remote file server process (RFS) on the standby site and can be stored as either standby redo logs (a new concept in Oracle9i Data Guard) or archived redo logs as in previous releases.
- Setting a Failure Resolution Policy: A failure resolution policy determines what actions will occur on a primary database when the last standby destination fails to archive redo logs. To set the highest level of data protection, guaranteed protection, place the primary database in `PROTECTED` mode using the following command:

```
ALTER DATABASE SET STANDBY DATABASE PROTECTED;
```

When this statement is used, the primary database is protected against data loss and divergence. If connectivity between the primary and standby database should now be lost, the primary database will shut down. When using this level of data protection, standby databases must have two or more standby redo log groups. Also, one or more primary database archive redo log destinations must have `LGWR` and `SYNC` attributes specified. The functionality of the `AFFIRM` attribute is implicitly set. You can revert to a mode that allows data divergence by mode using the following statement:

```
ALTER DATABASE SET STANDBY DATABASE UNPROTECTED;
```

Data Availability Mode Configuration Matrix

Data Availability Mode	Log Writing Process	Network Trans Mode	Disk Write Option	Redo Log Reception Option	Failure Resolution Option
Guaranteed	LGWR	SYNC	AFFIRM	Stb Rdo	Protect
Instant	LGWR	SYNC	AFFIRM	Stb Rdo	Unprotect
Rapid	LGWR	ASync	NOAFFIRM	Stb Rdo	Unprotect
Delayed	ARCH	ASync	NOAFFIRM	Arc Rdo	Unprotect

ORACLE

5-16

Copyright © Oracle Corporation, 2001. All rights reserved.

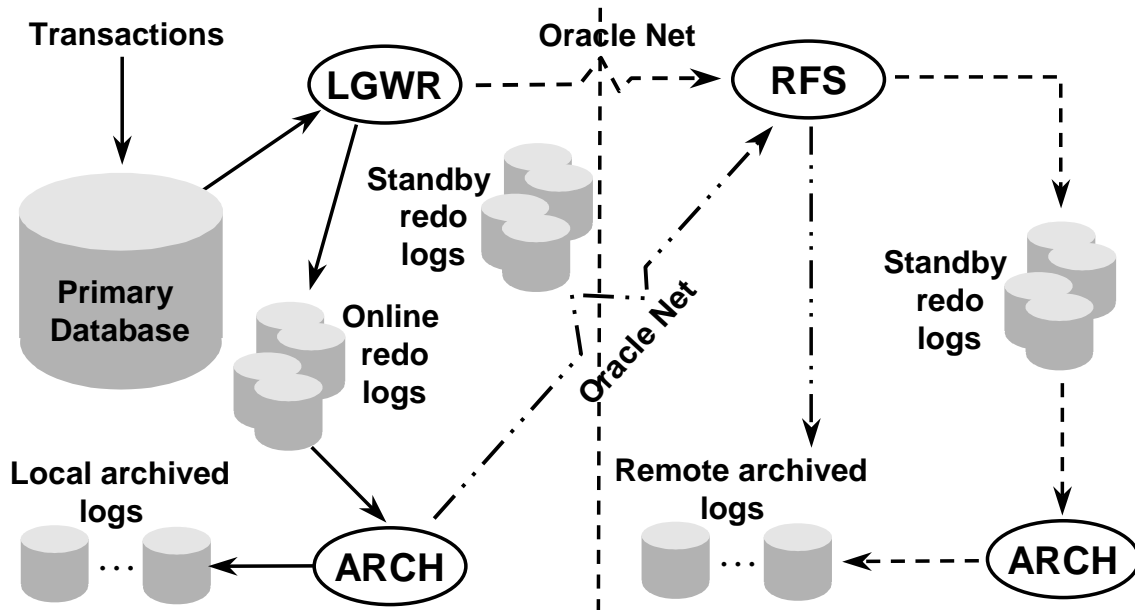
Data Availability Mode Configuration Matrix

Based on the previous slide, the above matrix shows you what attributes must be specified depending on the chosen data availability mode:

- Guaranteed and Instant modes have the best guarantee of data availability on the destination database, but the highest performance effect on the primary database.
- Rapid mode has a reasonable degree of data availability on the destination database, but a much decreased performance effect on the primary database.
- Delayed mode has the lowest degree of data availability on the destination database, and a slightly decreased performance effect on the primary database.

Note: Every combination between LGWR, ARCH, SYNC, ASync, AFFIRM, NOAFFIRM should be possible depending on your performance goals. In the above matrix, « Stb Rdo » stands for standby redo log files, and « Arc Rdo » stands for archived redo log files.

Redo Log Reception Possibilities



ORACLE

5-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Redo Log Reception Possibilities

Redo logs information transferred from the primary database are received by the remote file server process (RFS) on the standby site and can be stored as either standby redo logs or archived redo logs as shown in the above slide.

Oracle9i Data Guard introduces the concept of standby redo logs. Standby redo logs form a separate pool of log file groups.

Standby redo logs must be archived before the data can be applied to the standby database. The standby archival operation occurs automatically, even if the standby database is not in ARCHIVELOG mode. However, the archiver process must be started on the standby database.

Because of this additional archival operation, using standby redo logs may cause the standby database to lag further behind the primary database than when archiving directly from the primary database to standby destinations. However, the use of standby redo logs ultimately improves redo data availability on the standby database.

Note: Standby redo logs created on a primary database are not used until the database assumes the standby role. However, Oracle Corporation recommends that you create standby redo logs so that the primary database can switch roles easily and quickly without additional DBA intervention.

Also, Standby redo logs are required to implement a no-data-loss disaster recovery solution. Especially, if primary database failure occurs, redo data already written to standby redo logs can be fully recovered. See later in this lesson how to create and maintain standby redo log files.

Oracle9i: New Features for Administrators 5-17

Creating Standby Redo Logs

- Created using the **ADD STANDBY LOGFILE** clause of the **ALTER DATABASE** statement (**V\$STANDBY_LOG**, **V\$LOGFILE**)
- The minimum configuration: same number of groups as the primary database
- For best sizing, look at RFS trace files and alert log
- Consider the following parameters: **MAXLOGFILES**, **MAXLOGMEMBERS**, **LOG_FILES**

```
ALTER DATABASE ADD STANDBY LOGFILE  
( '/dbs/log4a.rdo', '/dbs/log4b.rdo' ) SIZE 10M;
```

```
ALTER DATABASE ADD STANDBY LOGFILE MEMBER  
' /dbs/log4c.rdo' TO GROUP 4;
```

ORACLE

Creating Standby Redo Logs

Standby redo logs are created using the **ADD STANDBY LOGFILE** clause of the **ALTER DATABASE** statement. Additional standby log group members can be added later to provide another level of reliability against disk failure on the standby site.

The best way to determine the appropriate number of standby redo log groups for a database instance is to test different configurations. The minimum configuration has the same number of groups as the primary database. The optimum configuration has slightly more groups than the primary database. In some cases, a standby database instance may require only two groups. In other situations, a database may require additional groups to guarantee that a recycled group is always available to receive redo information from the primary database. During testing, the easiest way to determine if the current standby log configuration is satisfactory is to examine the contents of the RFS process trace file and the database alert log. If messages indicate that the RFS process frequently has to wait for a group because an archival has not completed, add more standby groups.

Before setting up or altering the configuration of the standby redo log groups, consider the above parameters that can limit the number of standby redo log groups at the primary database creation time and standby instance startup time.

Note: If the primary database is operating in protected mode, and a standby redo log cannot be allocated, the primary database instance will be shut down immediately. Therefore, be sure you allocate an adequate number of standby redo logs. You can monitor standby redo logs using the new **V\$STANDBY_LOG** fixed view as well as **V\$LOGFILE**. A standby redo log group number cannot be the same as an existing online redo log group number. Also note that standby redo logs can be added on the primary as well as the standby databases.

Setting a Failure Resolution Policy

- **Determines what actions will occur on a primary database when all standby destination fails to archive redo logs:**
 - **PROTECTED:** Shut down the primary instance if connectivity is lost. No data loss and no divergence
 - **UNPROTECTED:** Allows data divergence
- **Use the alter database set command:**

```
ALTER DATABASE SET STANDBY DATABASE PROTECTED;
```

```
ALTER DATABASE SET STANDBY DATABASE UNPROTECTED;
```

```
SELECT STANDBY_MODE FROM V$DATABASE;
```

ORACLE

5-19

Copyright © Oracle Corporation, 2001. All rights reserved.

Setting a Failure Resolution Policy

A failure resolution policy determines what actions will occur on a primary database when the last standby destination fails to archive redo logs. To set the highest level of data protection, which is guaranteed protection, place the primary database in **PROTECTED** mode using the **SET STANDBY DATABASE** clause of the **ALTER DATABASE** statement as shown in the following example (note that database must be mounted **EXCLUSIVE** and not open for this operation):

```
ALTER DATABASE SET STANDBY DATABASE PROTECTED;
```

When this statement is used, the primary database is protected against data loss and divergence. If connectivity between the primary and standby database should now be lost, the primary database will shut down. When using this level of data protection, standby databases must have two or more standby redo log groups. Also, one or more primary database archive redo log destinations must have **LGWR** and **SYNC** attributes specified. The functionality of the **AFFIRM** attribute is implicitly set. You can revert to a mode that allows data divergence by placing the primary database in **UNPROTECTED** mode using the following statement:

```
ALTER DATABASE SET STANDBY DATABASE UNPROTECTED;
```

Note: You can determine the current status of your database by looking at the **STANDBY_MODE** column of the **V\$DATABASE** fixed view.

Finishing Managed Recovery

- If LGWR is sending redo log blocks to the Standby Database, then you are using Standby redo log files.
- In order to activate the Standby Database, you can:
 - Force its activation

```
ALTER DATABASE ACTIVATE STANDBY DATABASE  
SKIP STANDBY LOGFILE;
```

- Apply all committed transactions

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;  
RECOVER MANAGED STANDBY DATABASE FINISH;
```

- Register Standby redo logs and FINISH recovery

```
ALTER DATABASE REGISTER STANDBY LOGFILE  
'/standby/arch_dest/*.arch';
```

ORACLE

Finishing Managed Recovery

If you attempt to activate the standby database when there are standby log files that have not been recovered by the RECOVER MANAGED STANDBY DATABASE FINISH statement, an appropriate error will be signaled. If you want to override this behavior (intentional data loss), the following statement may be used:

```
ALTER DATABASE ACTIVATE STANDBY DATABASE SKIP STANDBY  
LOGFILE;
```

In case of an emergency at the primary database site, and you have access to the current online log, you should attempt to archive it, and then transfer and apply all available archived redo logs to the standby database.

If you do not use log transport services, manually copy the archivelog over to the standby database and use the REGISTER STANDBY LOGFILE command to identify the archivelog to the standby database. Use an appropriate operating system utility for transferring binary data. For example, enter:

```
$ cp /oracle/arc_dest/*.arc /standby/arc_dest
```

Then, register the archivelogs on the standby database. For example, enter the following SQL command:

```
ALTER DATABASE REGISTER STANDBY LOGFILE  
'/standby/arch_dest/*.arch';
```

Complete the recovery on the standby database, for example:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
```

Physical Standby Database Failover

- Previously, this was the only supported way to move the primary processing role to the standby.
- Failover was often only performed as a result of an unplanned outage of the primary.
- The primary had to be discarded and could not be used as the new standby.
- This was due to a `resetlogs` operation performed during the `ACTIVATE STANDBY` command.
- The system was at risk during the time taken to create a new standby.
- In Oracle9i, failover may still be used for catastrophic events.

ORACLE

5-21

Copyright © Oracle Corporation, 2001. All rights reserved.

Physical Standby Database: Failover

During a failover, the primary role is moved to one of the standby databases. The database that started as the primary cannot be used as a new standby database. This must be completely rebuilt using backups and logs from the new primary (old standby) database.

The normal failover process that forces database recreation occurs when all of the available archive logs are applied to the standby database, and the standby is opened as the primary using the following SQL command:

```
alter database activate standby database;
```

When this command is executed, a `resetlogs` operation is performed automatically on the standby and the new logs cannot be applied to the old primary.

The need to create a new standby database exposes a significant time interval of risk that is higher than the day-to-day operational mode because the standby has been removed as a failover option. The application must do without a standby's protection during the time the unavailable standby system is re-created and brought up-to-date with the new primary, which can take a day or two or even longer.

The biggest impact of this problem is that voluntary failover to the standby was performed with great reluctance because of the cost and risk of getting processing back to the original primary site. Failover was often reserved for nearly catastrophic events.

Physical Standby Database Graceful Switchover

- In Oracle9i, primary and standby database can continue to alternate roles.
- This is a planned operation, unlike failover.
- No need to recreate a new standby every time the switch is performed.
- Switchover is possible if all these conditions are met:
 - The primary performs a graceful shutdown
 - The archive logs are available
 - The primary's online logs are available

ORACLE

5-22

Copyright © Oracle Corporation, 2001. All rights reserved.

Physical Standby Database: Graceful Switchover

Switchover is possible only if the standby and its primary database can be brought to the same point in time. The prerequisites of initiating switchover include:

- The primary database has been shut down in an orderly fashion; that is, SHUTDOWN NORMAL, SHUTDOWN TRANSACTIONAL or SHUTDOWN IMMEDIATE.
- All archive logs required to bring the standby to the primary's point in time are available.
- The primary database's online redo logs are available and intact, as are its data files and control files.

Note: When using Real Application Clusters, only one instance is allowed to perform the switchover operation. All other instances must be shut down prior to performing the switchover operation.

Physical Standby Database Graceful Switchover

- The cost and risk of moving processing to the standby are now reduced.
- The primary becomes the new standby.
- Switchover becomes a useful solution to many problems:
 - Faster identification of difficult-to-diagnose problems
 - Reduction in planned downtime for hardware maintenance, patch installs and operating system upgrades
 - Protection against logical and physical corruptions

ORACLE

5-23

Copyright © Oracle Corporation, 2001. All rights reserved.

Physical Standby Database: Graceful Switchover (continued)

Previously, customers had to implement complicated steps or write their own switchover scripts. The manual switchover process requires DBA skills and knowledge of recovery, as well as the resources to test the implementation.

By automating switchover, downtime is minimized. By drastically reducing the cost and risk of moving production processing to a standby, these sites may now apply switchover to many more problems, executing switchover much more easily. This results in:

- Faster identification of difficult-to-diagnose problems. Like an accident or crime scene where bystanders are kept off to protect the evidence, database users are gracefully moved to the standby database. This allows the database administrators to perform diagnostics in a relatively quiet environment.
- Reduction in planned down time for operating system and hardware upgrades.
- Protection against logical and physical corruptions. Physical corruptions are caused by operating system, device drivers, or disks. Usually they are not propagated in the log stream.

Database Switchover Steps

The following steps are required to switchover:

1. End read or update activity on the primary and standby databases.
2. Prepare the primary database for switchover:

```
SELECT SWITCHOVER_STATUS FROM V$DATABASE;
```

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL  
STANDBY;
```

3. Shut down and start up the former primary instance without mounting the database.
4. Mount the former primary database in the standby database role.

ORACLE

Database Switchover Steps

- Step 1: Exclusive database access is required by the DBA before beginning a switchover operation. Ask users to log off the primary and standby databases and close all open sessions except the SQL*Plus session from which you are going to issue the switchover command.
- Step 2: On the primary database, to verify whether or not it is possible to perform a switchover operation, refer to the SWITCHOVER_STATUS (TO STANDBY) column of the V\$DATABASE fixed view before executing the following command: ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY; This command does the following:
 - Closes the primary database
 - Archives any unarchived log files and applies them to the standby database
 - Adds an end-of-redo marker to the header of the last log file being archived
 - Creates a backup of the current control file, and Converts the current control file into a standby control file
- Step 3: Execute the following command on the former primary database: SHUTDOWN NORMAL; and then STARTUP NOMOUNT; (Before ensure that the initialization parameters reflect the changes in database role).
- Step 4: Execute the following command on the primary database: ALTER DATABASE MOUNT STANDBY DATABASE;

Database Switchover Steps

5. Prepare the former standby database to switch to the primary database role:

```
SELECT SWITCHOVER_STATUS FROM V$DATABASE;
```

```
ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL  
PRIMARY;
```

ORACLE

5-25

Copyright © Oracle Corporation, 2001. All rights reserved.

Database Switchover Steps (continued)

- Step 5: To verify whether or not it is possible to perform a switchover operation, refer to the SWITCHOVER_STATUS (TO PRIMARY) column of the V\$DATABASE fixed view before executing the following command on the former standby database: ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL PRIMARY; This command does the following:

- Makes sure the last log file has been received and applied through the end-of-redo marker
- Closes the database if it has been opened for read-only transactions
- Converts the standby control file to the current control file

Note that the standby database must have been in managed recovery mode prior to starting the switchover operation, so that the primary database switchover operation request can be coordinated. If managed recovery was not active, or the primary database switchover notification was unable to be processed, the switchover operation will not be allowed to proceed. If the SWITCHOVER_STATUS column shows SWITCHOVER PENDING, this indicates that the switchover notification has been received from the primary database, but not yet processed. You need to manually place the standby database in managed recovery mode, so that the primary database switchover notification can be processed by the standby database.

Database Switchover Steps

6. Shut down the database.
7. Start up the database in the primary role.
8. Put the standby database in managed recovery mode.
9. Start archiving logs from the primary database to the standby database.

ORACLE

5-26

Copyright © Oracle Corporation, 2001. All rights reserved.

Database Switchover Steps (continued)

- Step 6: Execute the following command on the former Standby database:
`SHUTDOWN ;`
- Step 7: Execute the following command on the former Standby database:
`STARTUP ;` Be sure all the necessary initialization parameters point to the new standby database.
- Step 8: Execute the following command on the new standby database to place it in managed recovery mode: `ALTER DATABASE RECOVER MANAGED STANDBY DATABASE ;`
- Step 9: If not already done through initialization parameters, start archiving logs on the new primary database. You can issue the following commands:
 - `ALTER SYSTEM ARCHIVE LOG START ;`
 - `ALTER SYSTEM SWITCH LOGFILE ;`

Automatic Recovery of Log Gaps

- **A log gap occurs when the standby database is unable to process the next archived log.**
- **Gaps may occur:**
 - **At initial startup because the standby is not yet activated**
 - **During normal operation as a result of a temporary network problem**
- **Gaps are automatically detected.**
- **Missing logs are automatically fetched and applied.**
- **Archived logs can be fetched from a number of different servers.**

ORACLE

5-27

Copyright © Oracle Corporation, 2001. All rights reserved.

Automatic Recovery of Log Gaps

Using Oracle8i Standby Database, a number of everyday situations create the possibility that the standby database is unable to apply the next archive redo log. At initial startup, the logs in the gap are those created after the standby was created but before the standby is operational. In the general case, the primary has created them, but the standby has not received them because log shipping is not operational. During normal operations, this condition occurs when the standby database is activated, or as a result of WAN outages.

The Oracle9i eliminates the need for operator intervention. Any gaps are automatically detected on the standby database, and the missing logs automatically fetched from the primary database and applied.

This automation is further extended by the ability to fetch archive logs from any number of servers. At each standby, a list of servers that may have the missing archive logs are specified using TNS service names. If a connection to the primary database cannot be reestablished, the standby chains down the list, opens connections, and ships logs until it has secured all of the missing logs. Because a standby database can be configured to receive but not apply archive logs, the server used as an alternative log repository costs less than a complete standby database. It can receive and hold archive logs for a short period of time, perhaps a day, after which the logs can then be deleted, never having been applied. This avoids most of the storage and processing expense of another fully-configured standby database.

Automatic Recovery of Log Gaps Configuration

- **Managed recovery mode must be enabled.**
- **Set the following initialization parameters:**
 - **FAL_CLIENT** on the standby database
 - **FAL_SERVER** on the standby database
- **Those parameters correspond to TNS entries in the corresponding `tnsmanes.ora` files:**
 - **FAL_CLIENT** represents the standby database
 - **FAL_SERVER** represents the primary database or another standby database

ORACLE

5-28

Copyright © Oracle Corporation, 2001. All rights reserved.

Automatic Recovery of Log Gaps Configuration

In previous versions, to be able to place the standby database in managed recovery mode, you would first manually apply logs in the archive gap to the standby database.

In Oracle9i, you can set initialization parameters in order to automatically identify and resolve archive gaps as they occur.

The automatic recovery of log gaps is configured by enabling two processes through the initialization parameter file. The processes are:

- **Fetch archive log (FAL) client:** A background Oracle server process (the Remote File Server) that pulls archived redo log files from the primary site. The FAL client initiates and requests the transfer of archived redo log files automatically when it detects an archive gap on the standby database.
- **Fetch archive log (FAL) server:** A background Oracle server process (an Archiver process) that runs on the primary database (or another standby database used as staging area) and services the fetch archive log (FAL) requests coming from the FAL client. For example, servicing a FAL request might include queuing requests (to send archived redo log files to one or more standby databases) to an Oracle server that runs the FAL server. There can be multiple FAL servers running on the same primary database at any point in time; with a separate FAL server created for each incoming FAL request.

The DBA must specify the `FAL_CLIENT` and `FAL_SERVER` parameters on the standby database. These identify the TNS service names of the standby database and the primary database, respectively.

Archive Gaps Monitoring

```
SQL> SELECT thread#  
2      ,      low_sequence#  
3      ,      high_sequence#  
4 FROM    v$archive_gap;
```

THREAD#	LOW_SEQUENCE#	HIGH_SEQUENCE#
1	90	92

ORACLE

Archive Gaps Monitoring

To determine whether there is an archive gap, query the V\$ARCHIVE_GAP view. If an archive gap exists, the output of the query specifies the thread number and log sequence number of all logs in the archive gap. If there is no archive gap for a given thread, the query returns either no rows or an identical number in the LOW_SEQUENCE# and HIGH_SEQUENCE# columns.

In the above example, the gap is 90, 91, and 92 for thread 1.

Standby File Management

- **Add or drop of files in the standby database has been automated: the parameter `STANDBY_FILE_MANAGEMENT` enables this functionality if set to `AUTO`.**
- **If set to `AUTO`, `STANDBY_FILE_MANAGEMENT` prevents certain file manipulation on the standby database.**
- **Standby directory structure differences are much easier to maintain: the parameters `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` provide support for multiple pairs of filenames.**

ORACLE

5-30

Copyright © Oracle Corporation, 2001. All rights reserved.

Standby File Management

Using the new `STANDBY_FILE_MANAGEMENT` parameter, files added or dropped in the primary are automatically added or dropped in the standby during the normal course of applying changes. For this feature to be enabled, the parameter must be set to `AUTO`.

Furthermore, the parameters `DB_FILE_NAME_CONVERT` and `LOG_FILE_NAME_CONVERT` have been upgraded to provide support for multiple file (directory) transforms. As well as enabling the auto-file-add and -delete capability.

Certain operations are not allowed on the standby database if `STANDBY_FILE_MANAGEMENT` is set to `AUTO`: `ALTER DATABASE RENAME`, `ADD/DROP LOGFILE`, `ADD/DROP LOGFILE MEMBER`, and `CREATE DATAFILE AS`

Note: Non-Oracle Managed Files are still not automatically dropped on the standby. So, if the user drops a tablespace on the primary and asks that the files also be deleted (`DROP TABLESPACE ... INCLUDING DATAFILES`), the files are NOT deleted on the standby. Oracle Managed Files are discussed later in this course.

Background Managed Recovery Mode

- You can now create a background process to perform managed recovery:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE  
DISCONNECT FROM SESSION;
```

```
RECOVER MANAGED STANDBY DATABASE DISCONNECT  
FROM SESSION;
```

- The `FROM SESSION` clause is optional and is shown for clarification.

ORACLE

Background Managed Recovery Mode

You can now create a background process called MPR to perform managed recovery. This frees the foreground terminal session that you used to execute the managed recovery command.

If you want to start a detached server process and immediately return control to the user, add the `DISCONNECT [FROM SESSION]` option to the `RECOVER MANAGED STANDBY DATABASE` command syntax. Note that this does not disconnect the current SQL session.

Monitoring the Managed Recovery

```
SQL> SELECT process, status, thread#  
2      ,      sequence#, block#, blocks  
3      FROM    v$managed_standby;
```

PROCESS	STATUS	THREAD#	SEQUENCE#	BLOCK#	BLOCKS
MRP0	APPLYING_LOG	1	946	10	1001

ORACLE

5-32

Copyright © Oracle Corporation, 2001. All rights reserved.

Monitoring the Managed Recovery

To verify that you have correctly initiated the application of archived logs, query the V\$MANAGED_STANDBY fixed view on the standby database. This view monitors the progress of a managed recovery.

If you did not start a detached server process, you need to execute this query from another SQL session.

Updating the Standby at a Lag

- This feature delays the application of logs at the standby database.
- Normal log shipping proceeds as usual.
- The lag time is configurable by the DBA:

```
LOG_ARCHIVE_DEST_n = 'SERVICE=stby1 DELAY 30'
```

```
RECOVER MANAGED STANDBY DATABASE NODELAY;
```

- This reduces the chance of a mistake or corruption being propagated to the standby.

ORACLE

Updating the Standby at a Lag

The DELAY attribute of the LOG_ARCHIVE_DEST_n parameter specified on the primary database indicates that archive log files at the destination are not immediately available to the standby redo apply operation. The value of this optional attribute is expressed in minutes and specifies the time interval that must expire, after archive log transmittal has completed, before the archive log file can be processed. This attribute can be overridden at the standby during FINISH or emergencies.

The [NO]DELAY option of the RECOVER MANAGED STANDBY DATABASE command overrides any apply delay interval at the standby and applies redo as soon as the archive log is available (NODELAY), or applies it after the specified delay.

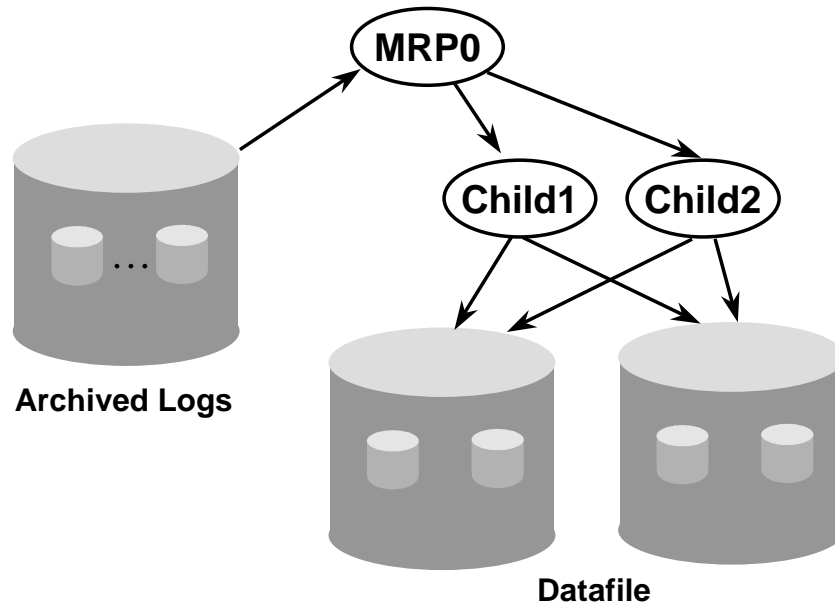
The FINISH option of RECOVER MANAGED STANDBY DATABASE implies NODELAY.

This is a scenario on how to use the DELAY attribute. Let's assume the standby database is running 30 minutes behind the primary (as shown on the above slide). This determines the reaction window the administrator will have to stop the propagation of user errors to the standby.

When a user error occurs, an assessment must be made of the extent of the damage and the effort required to recover from it. If the damage is relatively small, like deleting a wrong row, for example, and the rest of the database operations are not immediately affected by this error, tools like Flashback are well suited to deal with the problem.

However, when the extent of the damage is large and the time to recover is going to impact the production environment, it may be better to failover to the standby.

Parallel Recovery



RECOVER MANAGED STANDBY DATABASE PARALLEL 2;

ORACLE

5-34

Copyright © Oracle Corporation, 2001. All rights reserved.

Parallel Recovery

When running in managed recovery mode, the log apply services apply the changes generated on the primary database by many concurrent processes. Therefore, applying the archived redo logs on the standby database can take longer than the time it took to initially generate the changes on the primary database. By default, the log apply services use a single process to apply all of the archived redo logs sequentially. When using the parallel recovery option, several processes are able to apply the archived redo logs simultaneously. In general, using the parallel recovery option is most effective at reducing recovery time when several datafiles on several different disks are being recovered concurrently.

In a typical parallel recovery situation, one process is responsible for reading and dispatching archived redo logs. This is the dedicated managed recovery server process (either the foreground SQL session or the background MRP0 process) that begins the recovery session. The managed recovery server process reading the archived redo logs enlists two or more recovery processes to apply the changes from the archived redo logs to the datafiles.

You can execute parallel recovery using the **PARALLEL** option of the **RECOVER MANAGED STANDBY DATABASE** statement.

Miscellaneous Enhancements

- **New online redo log possibilities:**

- **Archive the CURRENT one without switching**

```
ALTER SYSTEM ARCHIVE LOG CURRENT NOSWITCH;
```

- **Archive them in mounted stage**

- **Archive them when using a controlfile backup**

```
ALTER SYSTEM ARCHIVE LOGFILE '/dbs/log2a.rdo'  
USING BACKUP CONTROLFILE;
```

- **Archive log repository**
- **Archive parameters can be set dynamically and archive attributes changes are allowed without the need to retype the entire string.**

ORACLE

5-35

Copyright © Oracle Corporation, 2001. All rights reserved.

Miscellaneous Enhancements

- In prior releases, it was possible to archive the current online redo log only when the instance was open. This command forced a log switch to occur. In Oracle9i, it is possible to:
 - Archive the current redo log without switching in open or mounted mode. In order to do that, use the NOSWITCH option of the ALTER SYSTEM ARCHIVE LOG CURRENT command. If done in open mode, the instance will automatically shutdown and the next startup will force a log switch.
 - Archive an online redo log based on the SCN (system change number) value when the database is mounted, but not open.
 - Archive an online redo log when a backup control file is being used. In previous releases, a current control file was required.
- The archive log repository is a stand-alone standby control file. This type of destination allows off-site archival of redo logs
- Two enhancements ameliorate the added complexity presented to those users who are not using the Oracle9i Data Guard Manager console for configuration and set-up: LOG_ARCHIVE_DEST_*n* initialization parameters can be set or cleared dynamically after the database has been started, and destination attributes can be changed without having to reenter the entire parameter string.

Miscellaneous Enhancements

- **REMOTE_ARCHIVE_ENABLE=TRUE | FALSE**
- **New attributes for LOG_ARCHIVE_DEST_#**
 - **ALTERNATE:** Automates failover to another destination
 - **MAX_FAILURE:** Maximum reopen failure count
 - **QUOTA_SIZE:** Maximum number of blocks that can be archived to a local destination
 - **DEPENDENCY:** Standby destinations depending on accessible local destinations
 - **Monitored through V\$ARCHIVE_DEST_STATUS**
- **You can specify up to ten archive destinations in Oracle9i. Prior versions allowed only up to five.**

ORACLE

5-36

Copyright © Oracle Corporation, 2001. All rights reserved.

Miscellaneous Enhancements (continued)

- When REMOTE_ARCHIVE_ENABLE initialization parameter is set to TRUE on both the primary and the standby database, the primary database is permitted to automatically archive online redo logs to remote archiving destinations, and the standby database is allowed to receive redo logs from the primary database. This is the default setting. When set to FALSE on the primary database, it indicates that the primary database is not permitted to automatically archive online redo logs to remote destinations. When set to FALSE on the standby database, it indicates that the standby is not permitted to receive redo logs from the primary database. This setting would be used in manual and testing recovery environments
- Many new attributes are required for the pre-existing LOG_ARCHIVE_DEST_# parameter. All log transport properties are enabled through it:
 - The new ALTERNATE attribute helps to ensure that logs make it to their destination if there is a failure by automating failover to another destination. The alternate destination can be on the same target system but using another communication link, or on another system, or even local to the primary. This attribute associates its destination with another destination. For example, in order to identify the name of the archive log destination that serves as the replacement if LOG_ARCHIVE_DEST_2 were to fail, could be specified by: ALTERNATE = LOG_ARCHIVE_DEST_3. This is valid when REOPEN=0 or MAX_FAILURE > 0 and failure count is exceeded (see next slide).

Miscellaneous Enhancements (continued)

Note: You can use the new V\$ARCHIVE_DEST_STATUS fixed view, and the new columns introduced in V\$ARCHIVE_DEST to monitor status errors of all the corresponding destinations attributes.

- How quickly this automatic failover occurs depends on other attributes. Using MAX_FAILURE, you can specify the maximum reopen failure count for the destination. By combining this with the existing REOPEN attribute, both the number and frequency of retries are completely under user control. For example, adding the following attributes to LOG_ARCHIVE_DEST_2: MAX_FAILURE = 10, and REOPEN = 10, LOG_ARCHIVE_DEST_2 fails over to LOG_ARCHIVE_DEST_3 after 100 seconds of failing retries.
- Another new LOG_ARCHIVE_DEST_ *n* attribute can be used with these to minimize the chances that an out-of-space condition stops archiving. The QUOTA_SIZE attribute identifies the maximum number of 512Kb blocks that can be archived to a local destination. The QUOTA_SIZE can be the size of the entire device, or any portion of it. A failover to another device could be planned after a certain amount of storage is used on the first destination. To ensure a fast switchover to the new destination, retry values would be to a minimal amount of retries over a small time period.
- With the DEPENDENCY attribute, a standby destination can be defined as being dependent upon the success or failure of an archival operation to another locally accessible destination (DEPENDENCY=LOG_ARCHIVE_DEST_1). The primary database archives a redo log locally and, upon successful completion, the archived redo log is immediately available to the standby database for media recovery. This does not require a physical remote archival operation for the standby destination. In this case, two destinations are used: one for local archiving, and another for archiving at the standby site. The standby destination is not valid unless the primary destination succeeds. Therefore, the standby destination has a dependency upon the success or failure of the local destination. Specifying a destination dependency can be useful in the following configurations:
 - The standby database and the primary database are on the same node. Therefore, the archived redo logs are implicitly accessible to the standby database.
 - Clustered file systems provide remote standby databases with access to the primary database archived redo logs.
 - Operating system-specific network file systems provide remote standby databases with access to the primary database archived redo logs.
- The number of log archive destinations has increased from 5 to 10 to provide more flexibility and to allow the co-existence of many different log clients: Physical Standby Database, LogMiner, Change Data Capture, and so on.

Summary

In this lesson, you should have learned how to:

- Position the Data Guard Broker and Data Guard Manager
- Specify the new attributes of the LOG_ARCHIVE_DEST_ *n* parameters
- Use the new ALTER DATABASE commands to change database roles and create standby redo logs
- Configure the FAL_CLIENT and FAL_SERVER parameters
- Use the DISCONNECT option of the managed recovery mode

ORACLE



Database Resource Manager Enhancements

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Set an active session pool for a consumer group.**
- **Automatically switch the consumer group of a running session.**
- **Set a maximum estimated execution time for each consumer group.**
- **Set a maximum limit on the total amount of undo generated per resource consumer group.**

ORACLE

Active Session Pool

- **The Database Resource Manager now allows a DBA to limit the amount of concurrent active sessions per resource consumer group by defining an active session pool.**
- **Benefits of an active session pool:**
 - **Allows DBAs to meet performance service level objectives by limiting the concurrent system workload**
 - **Reduces the number of servers taking resources in the system which avoid inefficient paging, swapping, and other resource depletion**

ORACLE

6-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Active Session Pool

Historically, the Oracle Database Resource Manager provided the mechanism to partition CPU resources. These resources were typically allocated at the beginning of a transaction or query and not freed until the transaction was committed, or query was finished. There were no means to free the resources associated with such operations until the operation was complete. Thus, newly arriving operations would either cause the system performance to be significantly impeded, or caused individual operations to error when unable to allocate a resource such as temporary space.

The active session pool feature allows the DBA to 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 since resource consumption is proportional to the number of active sessions. Once the active session pool is filled, the Database Resource Manager will queue all subsequent requests and run them only after existing active sessions complete.

The main goal of the active session pool is to reduce the number of servers taking resources in the system, thus avoiding inefficient paging, swapping, and other resource depletion (memory, temporary space, and so on.) resulting from attempting to run too many jobs simultaneously. In essence, it is an attempt to guarantee some minimum resources to an operation, and to establish limits on the resource consumer group.

Active Session Pool Mechanism

- **You can set an active session pool size per resource consumer group.**
 - **Active session pool is the maximum number of concurrently active sessions**
 - **Active session is defined as a session currently part of an active transaction, query, or parallel operation**
 - **Only one active session pool size is allowed per consumer group**
- **Once the active session pool is filled with active sessions, all subsequent sessions attempting to become active are queued.**
- **Individual parallel slaves do not count toward the number of sessions. The entire parallel operation counts as one active session.**

ORACLE

6-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Active Session Pool Mechanism

Queuing Method

Once the active session pool is filled with active sessions, the Database Resource Manager queues all subsequent sessions, attempting to become active until other active sessions complete or become inactive. With Oracle9i, there is only one queue per resource consumer group and the queuing method will be one of first in first out (FIFO) with a time-out.

The Queue is a simply memory structure. There is no view that shows the queue directly. To see some queue information you can use the following views:

- **V\$SESSION** - A new column called **CURRENT_QUEUE_DURATION**. If queued, it shows how long the session has been queued. It will be 0 (zero) if the session is not currently queued.
- **V\$RSRC_CONSUMER_GROUP** - A new column called **QUEUE_LENGTH**. This shows the number of sessions currently queued per consumer group.

Active Session Pool Parameters

You define the active session pool by setting the parameters **ACTIVE_SESS_POOL_P1** and **QUEUEING_P1**:

- **ACTIVE_SESS_POOL_P1** identifies the number of active sessions that establishes the resource consumer group's threshold.
- Default for **ACTIVE_SESS_POOL_P1** is 1000000
- **QUEUEING_P1** indicates how long in seconds any session will wait on the queue before aborting the current operation
- Default for **QUEUEING_P1** is 1000000

ORACLE

6-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Active Session Pool Parameters

ACTIVE_SESS_POOL_P1: Identifies the number of active sessions that establishes the resource consumer group's threshold, and therefore, its active session pool.

QUEUEING_P1: This optional parameter will indicate how long any session waits on the queue. If a session waits on the consumer group's queue for longer than the specified **QUEUEING_P1**, the operation will abort with an error.

Setting the Active Session Pool

- **Example:**

GROUP	ACTIVE SESSION POOL
OLTP	
BATCH	ACTIVE_SESS_POOL_P1 = 5 QUEUEING_P1 = 600

- **OLTP:** set no limit on concurrent active sessions
- **BATCH:** set to limit concurrent active sessions to 5. **QUEUEING_P1**, set to 600, aborts all operations waiting on the queue for more than ten minutes.

ORACLE

6-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Setting the Active Session Pool

Example

In the above example, the resource consumer group OLTP has no limit on the number of concurrent active sessions because the **ACTIVE_SESS_POOL_P1** parameter was not set. The resource consumer group BATCH has an **ACTIVE_SESS_POOL_P1** of **SIZE 5**. The **BATCH** group also has the **QUEUEING_P1** parameter set to 600 (ten minutes). All sessions waiting on the queue for more than ten minutes will abort with an error.

Maximum Estimated Execution Time

- The Database Resource Manager in Oracle9i can estimate the execution time of an operation proactively.
- A DBA can specify a maximum estimated execution time for an operation at the resource consumer group level using a new resource plan directive parameter.
 - `MAX_ESTIMATED_EXEC_TIME`: Maximum estimated time an operation can take
 - Operation will not start if estimate is longer than `MAX_ESTIMATED_EXEC_TIME`
- The benefit of this feature is the elimination of the exceptionally large job that uses too many system resources.
- The Default is 1000000.

ORACLE

6-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Maximum Estimated Execution Time

A DBA can define the maximum estimated execution time any operation can take at any given time by setting the resource plan directive `MAX_ESTIMATED_EXEC_TIME` parameter. After this parameter is set, the Database Resource Manager estimates the time a specific job will take. If the operation's estimate is more than the `MAX_ESTIMATED_EXEC_TIME` defined, then the operation will not start. This eliminates the exceptionally large job that would utilize too much of the systems resources.

If a resource consumer group has more than one plan directive referring to it, it may have more than one `MAX_ESTIMATED_EXEC_TIME`. The Database Resource Manager will then choose the most restrictive of all incoming values.

The estimated calculated time for a given statement is based on the statistics from the Cost Based Optimizer.

Automatic Consumer Group Switching

- **The Database Resource Manager automatically switches a session's consumer group based on the following resource plan directive parameters:**
 - **SWITCH_GROUP:** Group switched to. Default is `NULL`.
 - **SWITCH_TIME:** Active time in seconds. Default is `1000000`.
 - **SWITCH_ESTIMATE:** If value is `TRUE`, execution time estimate is used to decide whether to switch an operation even before it starts. Default is `FALSE`.
- **This feature can be used to limit the resources consumed by long running operations.**

ORACLE

6-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Automatic Change of Resource Manager Group

The Database Resource Manager will switch a running session to `SWITCH_GROUP` if the session is active for more than `SWITCH_TIME` seconds. Active means the session is running and consuming resources, not waiting idly for user input nor waiting for CPU cycles. After the session finishes its operation and becomes idle, it is switched back to its original group. If a resource consumer group has more than one plan directive referring to it, it may have more than one `SWITCH_GROUP` and `SWITCH_TIME`. The Database Resource Manager chooses the most restrictive of all incoming values.

If `SWITCH_ESTIMATE` is set to `TRUE`, the Database Resource Manager uses a predicted estimate of how long the operation will take to complete, to decide whether to switch a session before an operation even starts running. If this parameter is not set, the operation will just start running and will switch groups only if the other switch criteria are met.

A DBA can use this feature to manage the workload better by segregating long running batch jobs from short OLTP transactions. Batch jobs can be automatically assigned to consumer groups with lower resource allocation during day time to ensure good response time for OLTP users.

Undo Quota

- **New plan directive UNDO_POOL**
- **Limits the amount of undo space that can be used**
- **When exceeded, prevents DML (INSERT , UPDATE , DELETE)**
- **SELECT statements are still allowed.**
- **UNDO_POOL is specified in kilobytes.**
- **The default value for the UNDO_POOL is 1000000.**

ORACLE

6-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Undo Quota

The Database Resource Manager now allows a DBA to manage the undo space consumed by long running transactions, using a new resource plan directive parameter, UNDO_POOL.

It is defined as a quota of undo space per resource consumer group.

Whenever the total undo space is exceeded, no further INSERT , UPDATE or DELETE will be allowed until undo space is freed by another session in the same group or undo quota is increased.

If the consumer group's quota is exceeded during the execution of a DML statement, the operation will abort and return an error.

The UNDO_POOL limit can be used with Automatic Undo management and user defined rollback segments.

Changing Undo Quota

The undo quota can be changed anytime during database operation using the following resource plan directive:

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE  
(plan                => 'plan name',  
 group_or_subplan => 'consumer group name',  
 new_undo_pool    => 'new quota');
```

ORACLE

6-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Changing Undo Quota

If the pool is shrunk at or below the currently used amount by the group, the next session to request more undo will encounter an error, as will all subsequent sessions requesting more undo, until some undo is relinquished or the DBA raises the quota.

Modified Views to Support Database Resource Manager Extensions

- **New column added to V\$SESSION:**
 - **CURRENT_QUEUE_DURATION** : The current amount of time in seconds the session has been queued.
 - If not currently queued, value is zero.
- **New columns added to V\$RSRC_CONSUMER_GROUP:**
 - **QUEUE_LENGTH**: Number of sessions waiting on the queue
 - **CURRENT_UNDO_CONSUMPTION**: Current amount of undo consumed by consumer group

ORACLE

Modified Catalog Tables and Views

- **DBA_RSRC_PLANS** has a new column, **QUEUEING_MTH** defines the queuing resource allocation method for the plan, and **MAX_ACTIVE_SESS_TARGET_MTH** has been renamed to **ACTIVE_SESS_POOL_MTH**.
- **Seven new columns added to DBA_RSRC_PLAN_DIRECTIVES:**
 - **ACTIVE_SESS_POOL_P1**
 - **QUEUEING_P1**
 - **SWITCH_GROUP**
 - **SWITCH_TIME**
 - **SWITCH_ESTIMATE**
 - **MAX_EST_EXEC_TIME**
 - **UNDO_POOL**

ORACLE

Modified Catalog Tables and Views

The data dictionary view **DBA_RSRC_PLANS** has been changed. A new column **QUEUEING_MTH** has been added to show the method for queuing on the sessions. Currently there is only one method, First In First Out (FIFO), with a time out.

ACTIVE_SESS_POOL_MTH has one resource allocation method for the active session pool, which is an absolute number, as opposed to a percentage.

Other resource allocation methods are planned for future releases.

The following new columns were added to **DBA_RSRC_PLAN_DIRECTIVES**.

- **ACTIVE_SESS_POOL_P1**: First parameter for the active session pool resource allocation method
- **QUEUEING_P1**: First parameter for queuing resource allocation method.
- **SWITCH_GROUP**: Group to switch to.
- **SWITCH_TIME**: Amount of run time before session is automatically switched.
- **SWITCH_ESTIMATE**: TRUE if estimated execution time should be used for switch criteria.
- **MAX_EST_EXEC_TIME**: First parameter for the maximum estimated execution time.
- **UNDO_POOL**: Undo pool size for the consumer group.

An Example Using Several Resource Allocation Methods

- The example shown here could represent a plan for a database supporting a packaged ERP (Enterprise Resource Planning) or CRM (Customer Relationship Management).
- The work in such an environment can be highly varied.
- The goal is to give good response time to OLTP (Online Transaction Processing), while allowing batch jobs to run in parallel.

ORACLE

6-13

Copyright © Oracle Corporation, 2001. All rights reserved.

An Example Using Several Resource Allocation Methods

The example on the following page is based on this information:

Group	OLTP	Batch	Other_Groups
CPU Resource Allocation %	Level 1: 80%	Level 2: 100%	Level 3: 100%
Active Session Pool Parameters		Pool size: 5 Timeout: 600	
Automatic Switching Parameters	Switch to group: BATCH Switch time: 3 Use estimate: TRUE		
Max Estimated Execution Time		Time: 3600	
Undo Pool	Size: 200K		

An Example Using Several Resource Allocation Methods (continued)

```
BEGIN
/* Create the Pending area */
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
/* Create the Plan */
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'ERP_Plan', COMMENT
=> 'Resource plan/method for ERP Database');
/* Create the Resource groups, OLTP and BATCH */
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP =>
'OLTP', COMMENT => 'Resource consumer group/method for OLTP
jobs');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP =>
'BATCH', COMMENT => 'Resource consumer group/method for BATCH
jobs');
/* Create the Plan directive for OLTP */
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN =>
'ERP_Plan', GROUP_OR_SUBPLAN => 'OLTP', COMMENT => 'OLTP
sessions', CPU_P1 => 80, SWITCH_GROUP => 'BATCH', SWITCH_TIME
=>3, SWITCH_ESTIMATE => TRUE, UNDO_POOL => 200);
/* Create the Plan directive for BATCH */
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN =>
'ERP_PLAN', GROUP_OR_SUBPLAN => 'BATCH', COMMENT => 'BATCH
sessions', CPU_P2 => 100, ACTIVE_SESS_POOL_P1 => 5,
QUEUEING_P1 => 600, MAX_EST_EXEC_TIME => 3600);
/* Create the Plan directive for OTHER_GROUPS */
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN =>
'ERP_PLAN', GROUP_OR_SUBPLAN => 'OTHER_GROUPS', COMMENT =>
'mandatory', CPU_P3 => 100);
/* Validate and submit the plan */
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();

END;
```

Oracle Supplied Plans

- **SYSTEM_PLAN:** Plan to give system sessions priority
- **INTERNAL QUIESCE:** Plan to internally quiesce the system
- **INTERNAL_PLAN:** Only for testing

ORACLE

6-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Supplied Plans

These plans with a set of directives are created during database creation.

The `SYSTEM_PLAN` allocates the CPU to users in this way:

- Level 1 100% to `SYS_GROUP`
- Level 2 100% to `OTHER_GROUPS`
- Level 3 100% to `LOW_GROUP`

`SYS_GROUP` is the initial consumer group for the users `SYS` and `SYSTEM`. `LOW_GROUP` provided for your use, but no user will have this as an initial group. By default, all users have `OTHER_GROUPS` if not assigned to a group.

The `INTERNAL QUIESCE` plan is used when the database is put into the new quiesce state, which is described in a later lesson.

Summary

In this lesson, you should have learned how to:

- **Set parameters to define an active session pool.**
- **Set a maximum estimated execution time for each consumer group.**
- **Set parameters to automatically switch the consumer group of a running session.**
- **Set a maximum on the total amount of undo generated per resource consumer group.**

ORACLE



Online Operations

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Discuss new features designed to reduce planned downtime**
- **Explain online index rebuild new functionality**
- **Explain online functionality for index-organized tables (IOTs)**
- **Describe online table redefinitions**
- **Explain online analyze validate**
- **Describe database quiescing**
- **Describe the use of the server parameter file (SPFILE)**

ORACLE

Online Index Rebuild

- **Online index rebuild has now been extended to the following:**
 - Reverse key indexes
 - Function-based index
 - Key compressed indexes on regular tables
 - Key compressed indexes on index organized tables (including secondary indexes)
- **Updates are tracked in a journal table and later merged with the new index.**
- **There is no support for extensible (domain) indexes.**

ORACLE

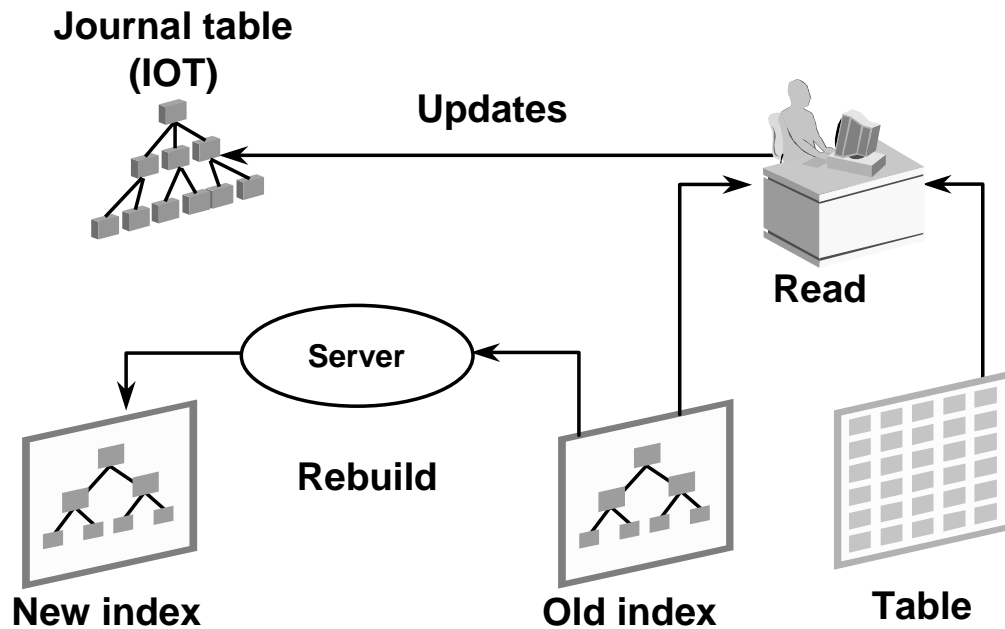
7-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Index Rebuild

There is currently no support for bitmap nor partitioned local and global indexes for online rebuilds.

Online Index Rebuild



ORACLE

7-4

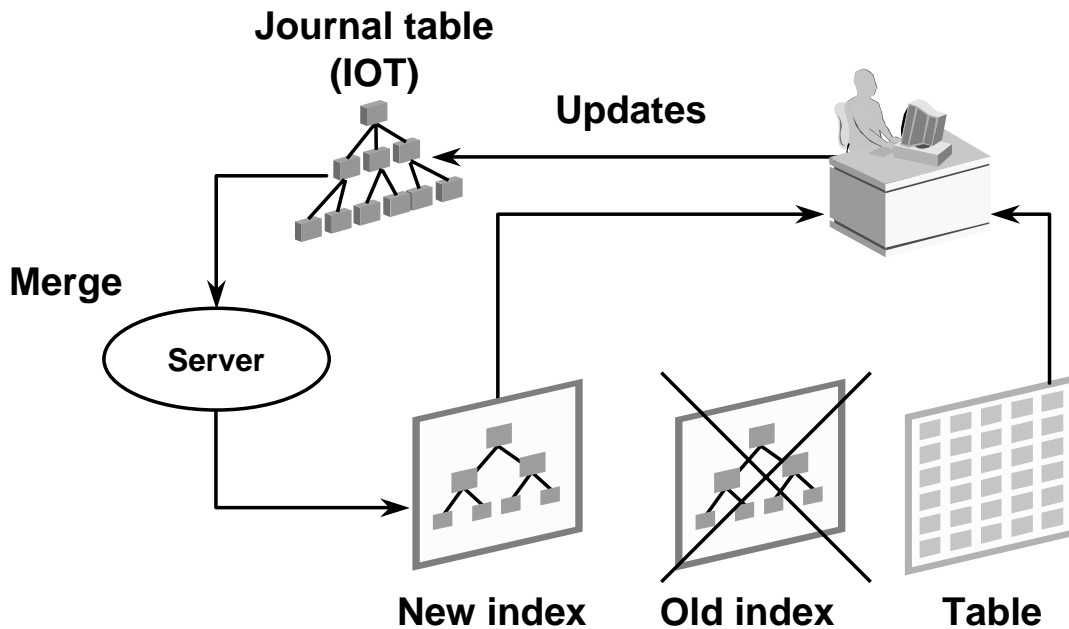
Copyright © Oracle Corporation, 2001. All rights reserved.

Online Index Rebuild (continued)

When the `ONLINE` keyword is specified as a part of an `ALTER INDEX` or `CREATE INDEX` command, a temporary index-organized journal table is created to record changes made to the base table. This journal table is created in the same tablespace as the index being altered or created.

While the server process is rebuilding the index, other users can continue to access the old index structure. Any updates to the old index during the rebuild operation are recorded in the journal table.

Online Index Rebuild



ORACLE

7-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Index Rebuild (continued)

When the server process has completed the rebuild operation, it merges the changes entered in the journal. This merge by which changed rows are incorporated into the new index is done while the table is still online.

This is accomplished by scanning the journal table and operating on a per row basis. Operations are committed every 20 rows. Locked rows are skipped. The Oracle server process may make multiple passes over the journal table to process previously locked rows.

Index-Organized Table High Availability Enhancements

Oracle9i extends this functionality to index-organized tables (IOTs):

- **Online create and rebuild of IOT's secondary indexes**
- **Online coalesce of IOT's primary indexes**
- **Online update of logical ROWIDs for secondary indexes on IOTs**
- **Online move of IOTs along with their OVERFLOW segment**

ORACLE

7-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Index Organized Table High Availability Enhancements

Online operations on index-organized tables and their indexes allow the table to be available for DML and query operations while the object redefinition is in progress. This is similar to support currently available for indexes on heap-organized tables.

One difference for index-organized tables relates to the COALESCE operation on the primary key index . Since index-organized table's primary key index cannot be directly altered, an ALTER TABLE <table_name> COALESCE operation has been introduced that coalesces the primary key index for an index-organized table. This coalesce works the same as it does for indexes on heap-organized tables.

Online Operations on IOTs Operations on Secondary Indexes

```
SQL> CREATE INDEX iot_secondary ON  
2 iot_employees (job_id) ONLINE;
```

```
SQL> ALTER INDEX iot_secondary REBUILD  
2 COMPUTE STATISTICS ONLINE;
```

```
SQL> CREATE INDEX iot_functional  
2 ON iot_employees  
3 (employee_id, salary + commission_pct)  
4 ONLINE;
```

```
SQL> ALTER INDEX iot_functional REBUILD ONLINE;
```

```
SQL> ALTER TABLE iot_employees COALESCE;
```

ORACLE

7-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Operations on IOTs: Operations on Secondary Indexes

Users can create B-Tree indexes on index-organized tables **ONLINE**. These can be normal or key-compressed B-tree indexes.

In the above examples, the following index organized table called **IOT_EMPLOYEES** is used:

```
CREATE TABLE iot_employees  
( employee_id,      first_name,      last_name,  
  email,            phone_number,  hire_date,  
  job_id,           salary,         commission_pct,  
  manager_id,       department_id, dn,  
  CONSTRAINT employees_pk PRIMARY KEY (employee_id))  
ORGANIZATION INDEX  
TABLESPACE data1 OVERFLOW TABLESPACE data1  
AS SELECT * FROM hr.employees;
```

Note: The **ONLINE** key word is not needed for the coalesce command.

Online Operations on IOTs

Online Update of Logical ROWIDs

- Secondary indexes on IOTs store a logical ROWIDs for each entry to improve performance.
- Since the logical ROWIDs can change, the value stored in the index are only a *best-effort guess*.
- These logical ROWIDs get stale over time.
- Now it is possible to update the stale logical ROWIDs online.
- The updates can also be done in parallel:

```
SQL> ALTER INDEX iot_secondary  
2 UPDATE BLOCK REFERENCES;
```

ORACLE

7-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Operations on IOTs: Online Update of Logical ROWIDs

This option improves performance for indexes on index-organized tables. Indexes on index-organized tables store logical ROWIDs, also called UROWIDs, to match the performance of an index on a conventional table. However, the logical ROWIDs can become stale. In such cases, an option under ALTER index allows fixing the logical ROWIDs online. This operation is performed in parallel if the object has a default parallel clause set.

Note: The ONLINE keyword is not required with the UPDATE BLOCK REFERENCES clause.

Restriction: You cannot combine this clause with any other clause of ALTER INDEX.

Online Operations on IOTs Online Move

- An IOT can be rebuilt to reduce fragmentation.
- The **MOVE** option of the **ALTER** command can be used to achieve this online.
- Now, the overflow data segment can also be moved online:

```
SQL> ALTER TABLE employees
      2 MOVE ONLINE    TABLESPACE data1
      3 OVERFLOW TABLESPACE data2;
```

ORACLE

7-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Operations on IOTs: Online Move

You can rebuild an index-organized table to reduce fragmentation by using the **MOVE** option of the **ALTER TABLE** command. The **MOVE** option rebuilds the primary key index B*-tree of the index-organized table but does not rebuild the overflow data segment, unless you specify the **OVERFLOW** clause explicitly, or you alter the **PCTTHRESHOLD** or **INCLUDING** column value as part of the **ALTER TABLE** statement.

Online Table Redefinition

- Occasionally, there is the need to reorganize a large heavily-used table.
- Previously, these redefinitions forced the table to be unavailable for the duration of the operation.
- From Oracle9i, table redefinitions can be performed online.
- The table to be redefined must have a primary key.

ORACLE

7-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition

In OLTP systems, occasionally there is the need to reorganize large heavily-used tables to improve the performance of queries or DML performed against these tables.

Until now, there was no simple mechanism to achieve this redefinition of the tables.

Moreover, while performing the redefinition of tables, the tables must be locked in shared mode which prevents any DML to the table during the time it is being redefined. For very large and frequently modified tables in OLTP systems, this may not be acceptable.

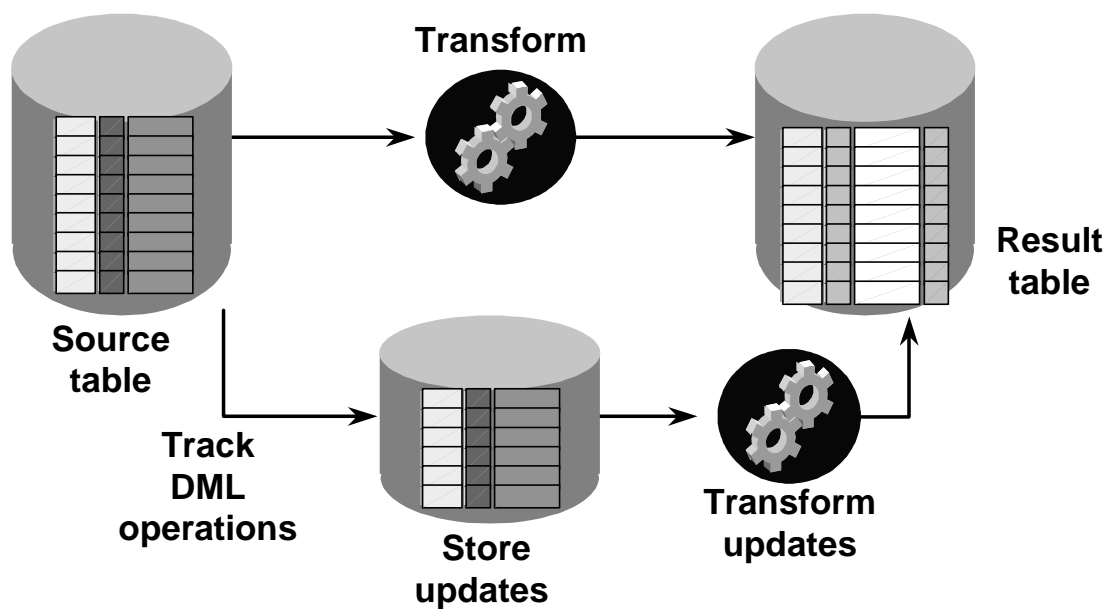
Additionally if the redefinition is achieved by exporting the table and importing it back, the table is unavailable during the entire process.

Online Table Redefinition Features

- A nonpartitioned table can be converted into a partitioned table and vice versa.
- The organization of a table can be changed from a heap based to IOTs and vice versa.
- Non-primary key columns can be dropped.
- New columns can be added to a table.
- Parallel support can be added or removed.
- Storage parameters can be modified.
- A column can be renamed.

ORACLE

Online Table Redefinition



ORACLE

Online Table Redefinition Syntax

1. **Determine if the table is a candidate for online Redefinition by calling the procedure**
`dbms_redefinition.can_redef_table.`
2. **Create an empty interim table with all the desired characteristics.**
3. **Start the Redefinition process calling**
`dbms_redefinition.start_redef_table.`
4. **Create any triggers, indexes, or constraints on the interim table.**
5. **Grant privileges on the interim table.**
6. **Finish the Redefinition process calling**
`dbms_redefinition.finish_redef_table.`

ORACLE

7-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition: Syntax

In order to perform an online redefinition of a table, the user must perform the following steps:

1. Invoke the `dbms_redefinition.can_redef_table` procedure to verify that the table can be redefined online. If the table is not a candidate, an error is raised. For example, a table without a Primary Key would get the following error:

```
ORA-12089: cannot online redefine table "SH"."SALES" with  
no primary key
```
2. Create an empty interim table in the same schema with all the desired attributes.
3. Call `dbms_redefinition.start_redef_table` to start the process. Provide as arguments: the name of the table to be redefined, the interim table name, and optionally, the column mapping and rollback segments. If no column mapping information is supplied, it is assumed that all the columns are in the interim table, with their names unchanged .

Online Table Redefinition: Syntax (continued)

4. The user can then create any triggers, indexes, and constraints on the interim table. The names must be different from the ones used in the original table. Any referential constraints involving the interim table should be created DISABLED. These constraints are automatically enabled on the completion of the online redefinition.
5. The user then executes the `dbms_redefinition.finish_redef_table` procedure to complete the redefinition of the table. During this procedure, the original table is locked in the exclusive mode for a very short period of time, which is independent of the amount of data in the original table. Also, as part of this procedure, the following occurs:
 - The original table is redefined to contain all the attributes, indexes, constraints, and triggers of the interim table
 - The referential constraints involving the interim table now involve the post-redefined table, and are enabled.
6. The user can optionally rename any indexes which were created on the interim table and which are now defined on the redefined table.

The following is the end result of the redefinition process:

- The original table is redefined with the attributes and features of the interim table.
- The triggers, grants, indexes, and constraints defined on the interim table after `dbms_redefinition.start_redef_table` and before `dbms_redefinition.finish_redef_table` are now defined on the post-redefined table. Any referential constraints involving the interim table before the redefinition process was finished now involve the post-redefinition table, and are enabled.
- Any indexes, triggers, grants, and constraints defined on the pre-redefined version of the original table are transferred to the interim table and are dropped when the user drops the interim table. Any referential constraints involving the original table before the redefinition now involve the interim table, and are disabled.
- Any PL/SQL procedures and cursors defined on the pre-redefined original table are invalidated. They are automatically revalidated whenever they are used next. This revalidation can fail if the shape of the table was changed as a result of the redefinition process.

Online Table Redefinition: Synchronization and Abort

- It is possible to periodically synchronize the interim table with the original one.
- The `dbms_redefinition.sync_interim_table` procedure performs the synchronization.
- This is recommended if there is a large amount of DML being processed between the start and finish of the redefinition.
- The `dbms_redefinition.abort_redef_table` procedure aborts the operation.

ORACLE

7-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition: Synchronization and Abort

Intermediate Synchronization

It is possible for a large amount of DML to be executed on the original table while the redefinition is taking place. In this case, it is recommended that the user periodically synchronizes the interim table with the original table by calling the `dbms_redefinition.sync_interim_table` procedure.

Calling this procedure reduces the time taken by `dbms_redefinition.finish_redef_table` to complete the redefinition process.

Note: The small amount of time that the original table is locked during `dbms_redefinition.finish_redef_table` is independent of whether `dbms_redefinition.sync_interim_table` has been called.

Abort

The user should call `dbms_redefinition.abort_redef_table` in the event that an error is raised during the redefinition process or the user wants to abort the redefinition process. This can be called any time after the `dbms_redefinition.start_redef_table` has been called and before `dbms_redefinition.finish_redef_table` is called.

Note: The interim table is not dropped after this procedure completes.

Online Table Redefinition: Example

**You want to reorganize the non-partitioned table
HR.EMP (EMPLOYEE_ID, FIRST_NAME, SALARY,
PHONE_NUMBER)**

- **Rename column EMPLOYEE_ID to EMPNO**
- **Multiply the SALARY column values by a factor of 1.10 and rename to SAL**
- **Drop column PHONE_NUMBER**
- **Add a column DEPTNO with default value 10**
- **Partition the table by range on EMPNO**

ORACLE

7-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition: Example

First the table must be checked to see if the table can be redefined.

```
SQL> EXECUTE dbms_redefinition.can_redef_table('HR','EMP');
```

Now create the interim table:

```
SQL> CREATE TABLE int_emp
2  (empno          NUMBER PRIMARY KEY,
3   first_name     VARCHAR2(20),
4   sal            NUMBER,
5   deptno         NUMBER DEFAULT 10)
6  (PARTITION emp1000 VALUES LESS THAN (1000) TABLESPACE tbs_1,
7   PARTITION emp2000 VALUES LESS THAN (2000) TABLESPACE tbs_2);
```

Now start the redefinition process:

```
SQL> EXECUTE dbms_redefinition.start_redef_table('HR', -
2  'EMP', 'INT_EMP', 'EMPLOYEE_ID EMPNO, FIRST_NAME' -
3  FIRST_NAME, SALARY*1.10 SAL');
```

At this point, create any triggers, indexes and constraints on interim table, INT_EMP.

Optionally, synchronize the interim table, INT_EMP. If you know that a large amount of DML has occurred on the original table, it is recommended that you periodically synchronize the interim table with the original table.

Online Table Redefinition: Example

```
SQL> EXEC dbms_redefinition.can_redef_table('HR','EMP');
```

```
SQL> CREATE TABLE int_emp          -- interim table
 2  (empno          NUMBER PRIMARY KEY,
 3   first_name     VARCHAR2(20),
 4   sal            NUMBER,
 5   deptno         NUMBER DEFAULT 10)
 6  (PARTITION emp1000 ..., PARTITION emp2000 ...);
```

```
SQL> EXECUTE dbms_redefinition.start_redef_table('HR', -
 2  'EMP', 'INT_EMP', 'EMPLOYEE_ID EMPNO, FIRST_NAME -
 3  FIRST_NAME, SALARY*1.10 SAL');
```

```
SQL> EXECUTE dbms_redefinition.sync_interim_table -
 2  ('HR','EMP','INT_EMP');
```

```
SQL> EXECUTE dbms_redefinition.finish_redef_table -
 2  ('HR','EMP','INT_EMP');
```

ORACLE

7-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition: Example (continued)

```
SQL> EXECUTE dbms_redefinition.sync_interim_table -
 2  ('HR','EMP', 'INT_EMP');
```

Then complete the redefinition:

```
SQL> EXECUTE dbms_redefinition.finish_redef_table -
 2  ('HR','EMP', 'INT_EMP');
```

Doing a describe on the tables shows that the interim table now looks like the original table:

```
SQL> describe hr.emp
Name                Null?    Type
-----
EMPNO               NOT NULL NUMBER
FIRST_NAME          VARCHAR2(20)
SAL                 NUMBER
DEPTNO              NUMBER
```

```
SQL> desc hr.int_emp
Name                Null?    Type
-----
EMPLOYEE_ID         NOT NULL NUMBER
FIRST_NAME          VARCHAR2(20)
SALARY              NUMBER
PHONE_NUMBER        VARCHAR2(20)
```

The interim table can now be dropped, since all the data is now in the new redefined table.

Online Table Redefinition Limitations

- **Primary key columns must be the same type and have the same number of columns.**
- **The following are not supported:**
 - **Tables without a primary key**
 - **User-defined data types**
 - **BFILE or LONG columns (LOBs are supported)**
 - **Tables with materialized view logs or materialized views defined on them**
 - **Horizontal subsetting of data**
- **New columns being added must not be declared as NOT NULL until the redefinition is complete.**

ORACLE

7-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition Limitations

Horizontal subsetting is the equivalent of putting a where clause on a select statement. The `dbms_redefinition.start_redef_table` has no facility for a WHERE clause. This means that this process will always get all rows from the original table.

The following is not a limitation but a suggestion. If you are redefining a table that was not partitioned into a table that is partitioned, it's a good idea to have the highest partition defined with MAXVALUES for the upper bound. This will avoid any errors if there are any values out of range.

Online Table Redefinition Limitations

- The redefinition must be done within the same schema.
- Snapshot or materialized view logs defined on the table are dropped as part of this redefinition.
- The following tables cannot be redefined:
 - Tables in the SYS or SYSTEM schema
 - An overflow table of an IOT
 - Materialized view container tables
 - Advanced queuing tables
 - Temporary tables
 - Clustered tables

ORACLE

7-19

Copyright © Oracle Corporation, 2001. All rights reserved.

Online Table Redefinition Limitations (continued)

Using DBMS_REDEFINITION package requires CREATE ANY TABLE, ALTER ANY TABLE, DROP ANY TABLE, LOCK ANY TABLE, SELECT ANY TABLE privileges, even if connected to the schema of the table you redefine. It is thus easier to use this package from a DBA account. You specify the schema of the table and intermediate table in the DBMS_REDEFINITION call.

Online Analyze Validate

- In Oracle9i you can now run the **ANALYZE VALIDATE STRUCTURE** command online.
- This command can run while DML changes are occurring on the object being validated.
- The new mode does not affect data availability:

```
SQL> ANALYZE TABLE employee  
2  VALIDATE STRUCTURE ONLINE;
```

ORACLE

Quiesce Database Overview

- **This is the ability to put the database into a partially available state.**
- **During this time, no ongoing non-DBA transactions, queries, or PL/SQL statements are allowed.**
- **Previously, this required the database to be shutdown and reopened it in restricted mode.**
- **Now, maintenance operations can be performed without forcing a shutdown.**
- **This is achieved by blocking new transactions.**

ORACLE

7-21

Copyright © Oracle Corporation, 2001. All rights reserved.

Quiesce Database - Overview

With this feature the database administrator can put the system into a *quiesced state*. During the time that the system is in quiesced state in the sense that, non-DBA transactions, queries, or PL/SQL statements are blocked from starting. In other words, database administrators are the only users that can proceed when the system is in quiesced state. This allows database administrators to perform certain actions that cannot be safely done when the system is not quiesced. These actions include:

- Actions that may possibly fail if there are concurrent user transactions or queries. For example, an attempt to change schema of a database table may fail if a concurrent transaction is accessing the same table.
- Actions whose undesirable intermediate effect could be seen by concurrent user transactions or queries. For example, to change the schema of a database table and update a PL/SQL procedure to a new version that use this new schema of the database table. The intermediate effect of the first step is an inconsistency between the schema of the table and the implementation of the PL/SQL procedure. This inconsistency could be seen by concurrent users that try to execute the PL/SQL procedure at the same time.

Without this feature, the DBA has to shut down the database and reopen it in restricted mode in many cases, which is a serious restriction for 24x7 systems. The quiesce database feature lessens such a restriction by providing database administrators the ability to perform these actions safely without shutting down database.

Quiesce Database – Overview (continued)

When a DBA issues a command to quiesce the database, all current user transactions and queries complete, releasing locks and other resources, although the users remain connected. At this point, the database becomes quiesced and the DBA can perform administrative tasks without interference.

Quiesce Database Benefits

- Users do not lose their sessions.
- There is no need for cache warmup because a shutdown is not required.
- Several maintenance operations benefit from this feature:
 - ALTER TABLE ...
 - CREATE OR REPLACE PROCEDURE ...
 - DROP TABLE

ORACLE

7-23

Copyright © Oracle Corporation, 2001. All rights reserved.

Quiesce Database: Benefits

Because any DDL command requires exclusive locks, if there are ongoing transactions on the objects being modified, the DDL statement fails. Also, any queries on objects that are being temporarily dropped, return errors. Modifications to objects in use by other sessions also cause inconsistencies in queries and stored procedures, because DDL statements implicitly commit themselves.

Defragmenting a table involves:

- Export the table
- Drop the table
- Import the table from export file

However, after the DROP, concurrent users that try to access the table could see its disturbing disappearance. Additionally, the DROP command could fail if other transactions have placed DML locks. All of these operations work properly in a quiesced database.

Note: The new quiesce database feature is not related to the quiescing from Advanced Replication.

Quiesce Database Syntax

- To put the database in quiesced state:

```
SQL> ALTER SYSTEM QUIESCE RESTRICTED;
```

- To return the database to its normal state:

```
SQL> ALTER SYSTEM UNQUIESCE;
```

- Both of these commands affect all the instances in a database, not just one.

ORACLE

7-24

Copyright © Oracle Corporation, 2001. All rights reserved.

Quiesce Database: Syntax

Two new commands have been introduced.

The `ALTER SYSTEM QUIESCE RESTRICTED` command puts all running instances into quiesced state. At the time when an `ALTER SYSTEM QUIESCE RESTRICTED` command is issued, there might be active non-DBA sessions in the system, such as inside a transaction, or a query. These sessions can proceed until they become inactive (the transaction or the query complete). Only after all these sessions become inactive does the `ALTER SYSTEM QUIESCE RESTRICTED` command return.

Additionally, once the `ALTER SYSTEM QUIESCE RESTRICTED` command is issued, even before it finishes, any attempt bring an inactive non-DBA session to an active status, such as issuing a SQL command, or starting a new query on that session, is blocked. This would appear to the user that their command is hung.

This two-fold approach ensures that once the `ALTER SYSTEM QUIESCE RESTRICTED` returns, the system is in quiesced state and the database administrator has complete isolation from concurrent non-DBA actions.

The `ALTER SYSTEM UNQUIESCE` command puts all running instances back into normal mode so that all blocked actions can proceed.

Quiesce Database Limitations

- **Resource Manager must be turned on for quiesce database to work:**
 - On every instance
 - Since instance startup
 - Without any interruption
- **When the database is being quiesced, opening a new instance results in an Real Application Clusters environment error.**
- **Backups taken during database quiesce have to use normal online backup procedures.**
- **Only users SYS and SYSTEM are considered DBA.**

ORACLE

7-25

Copyright © Oracle Corporation, 2001. All rights reserved.

Quiesce Database: Limitations

The implementation of this feature depends on the existence of the resource manager, and specifically, the maximum active sessions control feature of the resource manager. It is the resource manager that actually keeps track if sessions are active or not, blocks inactive non-DBA sessions from becoming active when the `ALTER SYSTEM QUIESCE RESTRICTED` command is issued, and frees blocked sessions when the system goes unquiesced.

The resource manager must have remained on in all open instances since their startup for the `ALTER SYSTEM QUIESCE RESTRICTED` command to succeed. If the resource manager is not turned on since startup in any open instances, or if the resource manager has been temporarily turned off (even if it is later turned on again) in any open instances, `ALTER SYSTEM QUIESCE RESTRICTED` command reports an error message. This is because getting the database into the quiesce state causes the `INTERNAL_QUIESCE` resource plan to be activated. This plan sets the `ACTIVE_SESS_POOL_P1` parameter to 0 (zero) for all groups except the `SYS_GROUP` which is the group that `SYS` and `SYSTEM` are in.

A recovery brings the system back to normal state in case of a crash while quiesced. Backup of files while the database is quiesced is considered a fuzzy backup, that is, normal online backup and restore procedures have to be observed.

For this release of Oracle9i, in the quiesce database context, a DBA is defined as user `SYS` or `SYSTEM`. Other users, regardless of the system privileges or roles they may have, are not allowed to proceed after the database is quiesced.

Viewing the Quiesce State of an Instance

Use the **ACTIVE_STATE** column of the **V\$INSTANCE** view to see what state the database is in:

ACTIVE_STATE	Description
NORMAL	Normal unquiesced state
QUIESCING	Being quiesced, still active non-DBA sessions
QUIESCED	Quiesced, no non-DBA sessions are active

ORACLE

7-26

Copyright © Oracle Corporation, 2001. All rights reserved.

Quiesce State

If the database is shut down while quiesced, when it is started up again, it will be in normal state.

Getting the database into the quiesce state is achieved by activating the **INTERNAL_QUIESCE** resource plan as described in the Database Resource Manager lesson. This plan sets the **ACTIVE_SESS_POOL_P1** parameter to zero for all groups except the **SYS_GROUP** which is the group that **SYS** and **SYSTEM** are in.

Server Parameter File (SPFILE)

- A new parameter file called SPFILE has been introduced to store instance parameters persistently by the server.
- Benefits include:
 - Ability to make changes to parameter values persist across shutdown and startup
 - Start the instance remotely without requiring a local parameter file copy

ORACLE

7-27

Copyright © Oracle Corporation, 2001. All rights reserved.

Server Parameter File

The SPFILE adds new functionality to the server for storing and managing its initialization parameters persistently in a server side disk file. The SPFILE provides the ability to make changes to the database persistent across shutdown and startup. This eliminates the need to manually update initialization parameters to make changes effected by ALTER SYSTEM statements persistent.

What Is an SPFILE?

- **A small binary file**
- **Maintained by the Oracle server**
- **Always resides on the server side**
- **Front end tools such as EM, SQL*Plus need not specify a parameter file if using the default SPFILE.**
- **The default server side location and name is platform specific; for example, for Unix:**
 - **`$ORACLE_HOME/dbs` is the location**
 - **`spfile<instance>.ora` is the default name**

ORACLE

7-28

Copyright © Oracle Corporation, 2001. All rights reserved.

What Is an SPFILE?

The SPFILE is a small binary file that cannot be browsed or edited using a text editor. The file is not meant to be modified manually and must always reside on the server side. Manually editing will corrupt the file. Your instance might be able to start using a corrupted SPFILE and if it is a running instance, it may even crash.

The SPFILE content is modified with `ALTER SYSTEM` commands, and content is displayed using the `SHOW PARAMETER SQL*Plus` command or selecting from `V$SPPARAMETER`.

By default the file is located in an operating system specific location, for example, `$ORACLE_HOME/dbs`, and has a default name in the format of `spfile<instance>.ora`. Once the file is created it is maintained by the Oracle server.

Creating a SPFILE

- **Created from an INIT.ORA file using the CREATE SPFILE command:**

```
CREATE SPFILE [= 'SPFILE-NAME' ]  
FROM PFILE   [= 'PFILE-NAME' ]
```

- **Where:**
 - **SPFILE-NAME:** Name of the SPFILE to be created
 - **PFILE-NAME:** Name of the INIT.ORA file to be used
- **Example:**

```
SQL> CREATE SPFILE FROM PFILE;
```

ORACLE

7-29

Copyright © Oracle Corporation, 2001. All rights reserved.

Creating a SPFILE

A SPFILE is created from an `init<instance>.ora` file using the `CREATE SPFILE` command. This statement can be executed when the database is in any state (IDLE, NOMOUNT, MOUNT or OPEN).

The next time you start up the database instance, you could direct Oracle to read the initialization parameters from the server parameter file.

The `CREATE SPFILE` command requires SYSDBA or the SYSOPER role to execute.

You can not recreate an SPFILE if the instance was started with the one you are trying to recreate. If you attempt this you will get:

```
ORA-32002: cannot create SPFILE already being used by the  
instance
```

Viewing the Parameter Settings

- The **V\$SPPARAMETER** view shows the contents of the **SPFILE**:
 - **SID**: SID for which parameter is defined
 - **NAME**: Parameter name
 - **VALUE**: Parameter value
 - **ISSPECIFIED**: Whether the parameter is specified in SPFILE
 - **ORDINAL**: Ordinal number of value if in list of strings
 - **UPDATE_COMMENT**: Last update comments
- The **UPDATE_COMMENT** column has been added to **V\$PARAMETER** and **V\$PARAMETER2**

ORACLE

7-30

Copyright © Oracle Corporation, 2001. All rights reserved.

Viewing Parameter Settings

There are several options for viewing the parameter settings. The new view **V\$SPPARAMETER** is created to provide the contents of the **SPFILE**.

A new column, **UPDATE_COMMENT**, which provides comments associated with **ALTER SYSTEM . . . COMMENT** command has been added to **V\$PARAMETER** and **V\$PARAMETER2**.

An example of **ORDINAL** is the **control_files** parameter. Since a database should have more than one control file this will show the order of the files listed after the **control_files** parameter. The output has been formatted.

```
SQL> select name,value,ordinal from v$spparameter
       2 where name ='control_files' ;
```

NAME	VALUE	ORDINAL
control_files	/databases/ed27/data/control01.ctl	1
control_files	/databases/ed27/data/control02.ctl	2
control_files	/databases/ed27/data/control03.ctl	3

Note: **SHOW SGA**, **V\$PARAMETER**, and **V\$PARAMETER2** display the currently in use parameter values.

Changing Parameter Values Within a SPFILE

- Use the **ALTER SYSTEM SET** to specify whether the change being made is temporary or persistent.
- Here **open_cursors** is being changed:

```
SQL> ALTER SYSTEM SET open_cursors = 400
      2 COMMENT='Test 1' SCOPE=spfile;

SQL> SELECT * FROM v$spparameter
      2 WHERE name = 'open_cursors';
```

SID	NAME	VALUE	ISSPEC	ORDINAL	UPDATE_COMMENT
*	open_cursors	400	TRUE	1	Test 1

- Use **ALTER SYSTEM RESET** to revert to default

ORACLE

7-31

Copyright © Oracle Corporation, 2001. All rights reserved.

Parameter Values Within an SPFILE

Using a server parameter file overcomes the limitation of the traditional way to use the **ALTER SYSTEM** command. Use the **SET** clause of the **ALTER SYSTEM** statement to set or change initialization parameter values. Additionally, the **SCOPE** clause specifies the scope of a change as follows:

- **MEMORY**: Changes of the parameter value are only in the currently running instance
- **SPFILE**: Changes of the parameter value are in the SPFILE only
- **BOTH**: Changes of the parameter value are in the currently running instance and the SPFILE

If a server parameter file was used to start up the database, **BOTH** is the default. If a parameter file was used to start up the database, **MEMORY** is the default.

For some dynamic parameters, you can also specify the **DEFERRED** keyword. When specified, the change is effective only for future sessions.

The **COMMENT** clause allows a comment string to be associated with the parameter update.

Reverting to Default Value

To revert to an undefined value, so the server default calculation is used, use the **RESET** clause of the **ALTER SYSTEM** statement. For example:

```
SQL> ALTER SYSTEM RESET LARGE_POOL_SIZE SCOPE=SPFILE SID='*';
```

is equivalent to deleting the parameter from an `init.ora` file.

STARTUP Command Behavior

```
SQL> STARTUP
```

- **Default SPFILE is used to start up the instance.**
- **If not found or unusable, the instance uses default server side INIT.ORA.**

```
SQL> STARTUP PFILE=$ORACLE_HOME/DBS/INITDBA1.ora
```

- **Specified INIT.ORA file is used to start up.**
- **The INIT.ORA file can optionally contain an indication to use an SPFILE.**

ORACLE

7-32

Copyright © Oracle Corporation, 2001. All rights reserved.

STARTUP Command Behavior

If you execute the STARTUP command without any PFILE specification, Oracle will search the default location (for example: \$ORACLE_HOME/dbs) in the following order, all of this is on the server side:

- SPFILE<\$ORACLE_SID>.ORA
- SPFILE.ORA
- INIT<\$ORACLE_SID>.ORA

You can also specify the SPFILE in a PFILE. This is the only way to start the instance with an SPFILE in a non default location. The entry in the PFILE would look like:

```
SPFILE = /database/startup/spfileoral.ora
```

To see which SPFILE was used for startup, you can look at the SPFILE parameter like any other initialization parameter (the “@” symbol is Oracle’s internal symbol for the \$ORACLE_SID, and the “?” is for \$ORACLE_HOME):

```
SQL> show parameter spfile
```

NAME	TYPE	VALUE
spfile	string	?/dbs/spfile@.ora

Exporting a SPFILE

- Contents of a SPFILE can be exported into an old-style PFILE.
- Example:

```
SQL> CREATE PFILE='newpfile.ora' FROM SPFILE;
```

- Created as a text file on the server side
- Requires the SYSDBA or SYSOPER role

ORACLE

7-33

Copyright © Oracle Corporation, 2001. All rights reserved.

Exporting a SPFILE

The contents of a SPFILE can be exported into an old-style PFILE in order to make modifications to the SPFILE by first exporting it, editing the output file, then recreating it. The PFILE is created as a text file on the server side. The command can be executed either before or after instance startup.

Syntax:

```
CREATE PFILE[ = 'PFILE-NAME' ] FROM SPFILE[ = 'SPFILE-NAME' ]
```

Exporting an SPFILE to a PFILE can also be used to create backups of the persistent parameter file.

Note: If you specify no names for the files, a platform-specific name is used for the initialization parameter file, and it is created from the platform-specific default server parameter file. For example in the above command on the slide, the file would be created in the \$ORACLE_HOME/dbs directory on many UNIX platforms.

Example of an Exported SPFILE

- The star (*) notation refers to all instances
- The file can be used for a startup.

```
...  
*.open_cursors=350  
*.processes=150  
*.remote_login_passwordfile='NONE'  
*.resource_manager_plan='SYSTEM_PLAN'  
*.sga_max_size=0  
*.shared_pool_size=8388608  
...
```

ORACLE

Migrating to a SPFILE

- FTP the `init.ora` file from the client side to the server side, if needed.
- Create a server parameter file using the **CREATE SPFILE** statement.

```
CREATE SPFILE [= 'SPFILE-NAME' ]  
FROM PFILE [= 'PFILE-NAME' ]
```

- Use the created **SPFILE** to start up the instance.

```
STARTUP
```

ORACLE

Summary

In this lesson, you should have learned how to:

- **Perform online operations on all types of indexes**
- **Perform online operations on index-organized tables (IOTs)**
- **Perform online table redefinitions**
- **Put the database in quiesced state**
- **Use the new Server Parameter File (SPFILE) for database startup**

ORACLE

Practice 7-1 Overview

This practice covers the following topic:

Redefine a table online:

- **Rename columns**
- **Add columns**
- **Drop columns**
- **Partition the table**

ORACLE

Practice 7-2 Overview

This practice covers the following topic:

Use of an SPFILE:

- **Create an SPFILE**
- **Examine the contents of the SPFILE**
- **Make changes to the parameters**
- **Create a PFILE from the SPFILE**

ORACLE

8

Segment Management (Part I)

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Use the automatic global index maintenance feature**
- **Understand the use of external tables**
- **Use the new LIST partitioning method**
- **Understand the basics of meta data Application Programming Interface**

ORACLE

Global Index Maintenance During Partition DDLs

- Prior to Oracle9i, when a partition DDL statement was issued against a partitioned table, all indexes corresponding to the data partition affected by the DDL were marked `UNUSABLE`.
- Because the rebuild is a costly operation, many users were reluctant to use global indexes.
- Oracle9i provides the capability to automatically update a global index during a DDL by way of an optional clause.

ORACLE

Maintaining Global Indexes During DDL

Partitioning allows very large tables to be broken in smaller partitions that can be individually managed. Indexes (global and local) can be defined on such tables. Oracle supports a number of DDLs that operate on and allow for maintenance of partitions. When such operations are issued, index partitions that correspond to the data partitions affected by the DDL are marked `UNUSABLE`. The DBA is then required to rebuild the indexes. This is not a problem for local indexes as the DBA only needs to rebuild the local index of the partition(s). Unlike local indexes, when a partition DDL is performed, even though the partition DDL might affect only a small fraction of the data in the table, the entire global index is marked `UNUSABLE` and must be rebuilt. This can be a very costly operation. Having the capability to automatically update the Global Indexes during the DDL eliminates this problem.

Benefits of Maintaining Global Indexes During DDL

The benefits to this new feature include:

- **Usability:** The partition DDL becomes more user friendly
- **Availability:** Makes the database more available
- **Manageability:** Global indexes are maintained in conjunction with the base table

ORACLE

8-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits of Maintaining Global Indexes During DDL

- **Usability:** This feature makes partition DDL much more user friendly. This is because the user does not need to look up the names of all UNUSABLE global indexes and then issue DDL statements to individually rebuild them. This feature replaces a two step process (partition DDL, rebuild entire index) with a one step process.
- **Availability:** Using this feature makes the database more available. If a user issues a partition DDL and specifies the UPDATE GLOBAL INDEXES clause, all global indexes are available throughout the operation. In contrast, if one did not use this feature, global indexes would be marked UNUSABLE. They would remain UNUSABLE until they were rebuilt. In practice, this would have a severe performance and availability impact on applications accessing this partitioned table. For this reason this feature should be used only in case the amount of data affected by the partition DDL operation is small compare to the entire index size.
- **Manageability:** Global indexes are maintained in conjunction with the base table. A user does not have to rebuild global indexes independently.

Maintaining Global Indexes During DDL

- The optional clause **UPDATE GLOBAL INDEXES** can be added to partition DDL syntax to allow valid global indexes to be updated during DDL operations.
 - **INVALIDATE GLOBAL INDEXES** is the default
- The clause is allowed between the partition specification clause and the parallel clause:

```
SQL> ALTER TABLE test
2 DROP PARTITION tsp
3 UPDATE GLOBAL INDEXES
4 PARALLEL (DEGREE 4);
```

ORACLE

Maintaining Global Indexes During DDL

- This option is available when executing the following partition DDL statements:
 - SPLIT, MERGE, ADD, COALESCE, MOVE, DROP, TRUNCATE, EXCHANGE
- The parallel clause is allowed in conjunction with the new syntax for DROP, TRUNCATE, and EXCHANGE when updating global indexes.
- This functionality supports regular B*-tree and function-based indexes.

ORACLE

8-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Maintaining Global Indexes During DDL

For EXCHANGE partition, only global indexes on the table whose partition is being exchanged will be updated. Indexes for the table being exchanged will continue to be invalidated. For example:

```
SQL> ALTER TABLE sales EXCHANGE PARTITION q1_2000  
2 WITH TABLE sales_q1_2000 UPDATE GLOBAL INDEXES;
```

The global indexes for table SALES are updated when partition q1_2000 is exchanged with table sales_q1_2000.

Maintaining Global Indexes During DDL

- **This functionality does not support the following:**
 - Index-organized tables
 - Domain indexes
 - Local indexes
- **Only indexes that are valid and in a `USABLE` state will be updated.**

ORACLE

8-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Maintaining Global Indexes During DDL

This functionality does not support local indexes. After a partition DDL, local indexes can be rebuilt by the command `ALTER TABLE MODIFY PARTITION...REBUILD UNUSABLE LOCAL INDEXES`. Using this command is more efficient.

Update or Rebuild Global Indexes?

- **Using global indexes updates:**
 - Partition DDL takes longer to complete because the global indexes that were previously marked invalid are now updated.
 - DROP, TRUNCATE, EXCHANGE are no longer fast dictionary operations, because a scan of all rows in the partition is needed.
 - Global index updates are logged; therefore, redo and undo logs will be generated.
- **Rebuilding the entire index may improve efficiency.**
- **Rebuilding the index allows for index reorganization.**

ORACLE

Update Versus Rebuild Global Indexes

Note: The degree of improvement in the efficiency depends on the number of rows impacted by the partition DDL command.

List Partitioning Overview and Benefits

- New partition method introduced in Oracle9i
- User controls how rows map to partitions
- The LIST partition method allows for the distribution of data, based on discrete column values.
- Unordered and unrelated sets of data can be grouped and organized together very naturally, using LIST partitioning.
- No relationship between partitions
- Ideal for columns that consist of discrete values
- Powerful data management capabilities

ORACLE

List Partitioning Overview and Benefits

The Partitioned Objects features has incrementally added new partition methods to the Oracle RDBMS over two releases: Oracle8 and Oracle8i. However, there are constantly emerging requirements generated by customers who cannot take full advantage of Oracle8i partition methods provided so far, because their data model does not dovetail Oracle partition methods.

Thus, Oracle9i adds a new partitioning model called LIST partitioning to the set of partition methods already being supported in the Oracle RDBMS. LIST method allows explicit control over how rows map to partitions. This is done by specifying list of discrete values for the partitioning column in the description for each partition. LIST partitioning is different from RANGE partitioning where a range of values is associated with a partition, and from HASH partitioning where the user has no control of the row-to-partition mapping. This partition method has been specifically added to model data-distributions that follow discrete values. This cannot be easily done with Oracle8i.

List Partitioning Example

```
CREATE TABLE locations (location_id, street_address,  
postal_code, city, state_province, country_id)  
PARTITION BY LIST (state_province)  
STORAGE(INITIAL 10K NEXT 20K) TABLESPACE tbs5  
(PARTITION region_east  
VALUES('MA','NY','CT','NH','ME','MD','VA','PA','NJ')  
STORAGE (INITIAL 20K NEXT 40K PCTINCREASE 50)  
TABLESPACE tbs8  
,PARTITION region_west  
VALUES('CA','AZ','NM','OR','WA','UT','NV','CO')  
PCTFREE 25 NOLOGGING  
,PARTITION region_south  
VALUES('TX','KY','TN','LA','MS','AR','AL','GA')  
,PARTITION region_central  
VALUES('OH','ND','SD','MO','IL','MI',NULL,'IA')  
);
```

ORACLE

8-10

Copyright © Oracle Corporation, 2001. All rights reserved.

List Partitioning Example

The details of LIST partitioning can best be described with an example. In this case we want to partition the LOCATIONS table by region, that is, we want to group states together according to their geographical location.

A row is mapped to a partition by checking whether the value of the partitioning column for a row falls within the set of values that describes the partition.

For example, the following rows will be inserted as follows

- (1500,'2011 Interiors Blvd', '99236','South San Francisco', 'CA', 'US') maps to partition region_west
- (5000,'Chemin de la Fanee', '13840','ROGNES', 'BR', 'FR') will not map to any partitions in the table and thus will fail with an ORA-14400 error. This partitioning method is useful when the partitioning column consists of bounded set of discrete values. However, in this release, it is not possible to define a catch-all partition (like MAXVALUE for range partitioning). This will be available in a later release of Oracle9i.

In the above example, the partitions with specified physical attributes will override the table-level defaults. However, any partition with unspecified attributes inherits their physical attributes from the table-level defaults.

Note: For formatting reasons, the column data types are not specified. Refer to the Oracle9i Sample Schema definition for more details on those data types.

List Partitioning Pruning

```
SQL> SELECT * FROM locations  
2 WHERE state_province = 'CA';
```

```
SQL> SELECT * FROM locations  
2 WHERE state_province IN ('CA','NY');
```

```
SQL> SELECT * FROM locations  
2 WHERE state_province <= 'AZ';
```

ORACLE

8-11

Copyright © Oracle Corporation, 2001. All rights reserved.

List Partitioning Pruning

One of the interesting things to note about LIST partitioning is that there is no apparent sense of ordering between partitions (unlike RANGE). Nevertheless, Oracle9i supports partition pruning for objects partitioned by LIST method for queries involving the following predicates on the partitioning key:

- Equality: Only the corresponding partition is accessed. Based on the LOCATIONS table and the first query above, only partition `region_west` is accessed.
- IN-LIST: Only the corresponding partitions are accessed. Based on the LOCATIONS table and the second query above, Oracle accesses only `region_west` and `region_east`.
- Range: Only partitions that contain literal values in their list that correspond to the range predicate are accessed. Based on the LOCATIONS table and the third query above, only partition `region_west` ('CA','AZ','NM','OR','WA','UT','NV','CO') and partition `region_south` ('TX','KY','TN','LA','MS','AR','AL','GA') are accessed.

ALTER TABLE ADD PARTITION

New values cannot exist in any of the other partitions.

```
SQL> ALTER TABLE locations
2  ADD PARTITION region_nonmainland
3  VALUES ('HI', 'PR')
4  STORAGE (INITIAL 20K NEXT 20K)
5  TABLESPACE tbs_3
6  NOLOGGING;
```

ORACLE

8-12

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER TABLE ADD PARTITION Example

In Oracle9i, syntax and semantics of this DDL statement is modified to support adding a single partition to a table partitioned using LIST method. This just means that the user is adding a new partition to the set of partitions of a table. However, there is no ordering among the partitions. The newly created partitions have the following characteristics:

- Newly added partitions have no data.
- A partition name, a set of literal values describing the partition value list, physical attributes, and logging attributes can be specified for such partitions.
- Every literal value in the set that describes the partition value list for a partition has to be a unique value specified for that object. If there is a duplicate entry, Oracle will return an error.
- If any physical attributes of such a partition are not specified, they are derived by combining default table-level attributes with default attributes of a tablespace in which the partition is placed.
- If a partition-name is not specified, a system-generated name of form SYS_P### is assigned to the partition.

ALTER TABLE ADD PARTITION Example (continued)

- Adding a partition to a table adds a corresponding index partition with the same value list to all local indexes defined on the table. Global indexes are not affected.
- ALTER TABLE ADD PARTITION to a list partitioned table locks the table in Exclusive mode(X) for the duration of the operation. This is a fast dictionary operation and no other DDL or DML is permitted on the table.

The above example adds a new partition to the LOCATIONS table partitioned by the LIST method. The example specifies some new physical attributes for this new partition and inherits the table-level default for the others.

ALTER TABLE MERGE PARTITION

Merge the contents of two partitions of a LIST partitioned table.

```
SQL> ALTER TABLE locations
  2  MERGE PARTITIONS
  3  region_northwest, region_southwest
  4  INTO PARTITION region_west
  5  PCTFREE 50 STORAGE(MAXEXTENTS 20);
```

ORACLE

8-14

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER TABLE MERGE PARTITION

The syntax of this operation remains unchanged for a LIST partitioned table compared to a RANGE partitioned table. But the semantics are modified to operate on tables partitioned using the LIST method. The statement can be used to merge the contents of any two arbitrary partitions of a table. The candidate partitions do not need to be adjacent, since LIST partitions do not assume any order for partitions. The resulting partition contains the union of the two sets of values from the two partitions being merged. Usage :

- If a name for the resulting partition is not specified, a name of form SYS_P# is to be used.
- Any two partitions can be merged.
- The resulting partition value list consists of a set of literal-values that represents the union of the set of literal-value partition-values-list for the two partitions being merged.
- The data in the resulting partitions consists of data from both the partitions.
- The tablespace in which the resulting partition is located, and the partition's attributes are determined by the table-level default attributes, except for those specified explicitly.
- The corresponding local index partitions will also be merged and the resulting local index partition will be marked Index Unusable.

ALTER TABLE MERGE PARTITION (continued)

- All Global indexes defined on the table will be marked Index Unusable. These include both partitioned and non-partitioned indexes.
- This will lock the table in SX mode and the partitions being merged in X mode.

The above example merges two partitions of a the `LOCATIONS` table partitioned using `LIST` method into a partition that will inherit all of its attributes from the table-level default attributes, except for `PCTFREE` and `MAXEXTENTS` which are specified in the statement.

If the value-list for:

- partition `region_northwest` was described as ('CA','OR','NV','UT') and
- partition `region_southwest` was described as ('AZ','NM','CO','WA')
- then, the resulting partition-value-list will contain the set that represents the union of these two partition-value-lists:
 - partition `region_west` will have value-list as ('CA','OR','NV','UT','AZ','NM','CO','WA')

Note: For this example, we suppose that at some point the initial partition `region_west` was split into `region_northwest` and `region_southwest`.

ALTER TABLE MODIFY PARTITION ADD VALUES

New values cannot exist in any of the other partitions:

```
SQL> ALTER TABLE locations
      2  MODIFY PARTITION region_south
      3  ADD VALUES ('OK', 'KS');
```

ORACLE

8-16

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER TABLE MODIFY PARTITION ADD VALUES

This is a new statement which can be used to *extend* the partition value list of an existing partition to contain additional literal values. The statement is used to describe the new set of literal values that are being added to an existing partition value list to extend it. All the new literal values being added must not have been previously specified in any of the partition's value list that describe the partition table.

The above example adds a new set of state codes ('OK','KS') to the existing partition list `region_south`.

Usage:

- The set of literal-values that are being added to the partition have to be unique within the set of literal values that describe partition value list, for all existing partitions. If duplicate values are found, Oracle will return an error.
- The partition literal value list for the corresponding local index partition is also naturally extended.
- Status (Index Unusable) of local and global index partitions are not affected by this operation.
- During this fast operation, a DML Share-Exclusive (SX) lock is acquired on the table and a DML Exclusive (X) lock is acquired on the partition being modified.

ALTER TABLE MODIFY PARTITION DROP VALUES

Only possible if you don't have corresponding rows:

```
SQL> DELETE locations
2  WHERE state_province in ('OK','KS');
```

```
SQL> ALTER TABLE locations
2  MODIFY PARTITION region_south
3  DROP VALUES ('OK','KS');
```

ORACLE

8-17

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER TABLE MODIFY PARTITION DROP VALUES

This is a new statement which can be used to *prune* the partition value-list of an existing partition to contain fewer literal values. This operation removes a set of literal-values from an existing set of literal values that describes the partition's value list. It is important to note that this operation expects no rows to exist in the partition for the literal values being dropped. However, it checks for the existence of rows in the partition that correspond to the literal values being dropped and fails, with an error message, if any such qualifying rows are found. Thus, the user must drop the corresponding rows first.

The above statement drops a set of state codes ('OK','KS') from the existing `region_south` partition value-list. The operation is always run with validation, which means it checks to see if any rows exist in the partition that correspond to the set of values being dropped. If any such rows are found, Oracle returns an error message and the operation fails. The user is supposed to issue a `DELETE` statement to delete the corresponding rows from the table. This operation also drops the corresponding values from the dictionary.

Usage:

- The set of literal-values that are being dropped from the partition have to be a subset of the set of literal values that describe the current partition value list. Oracle will return an error message otherwise.
- This operation cannot be used to drop all values that correspond to a partition. One should use `ALTER TABLE DROP PARTITION` instead.

ALTER TABLE MODIFY PARTITION DROP VALUES (continued)

- The dictionary is also updated to reflect the new partition-value-list for the partition being modified.
- During the operation, a DML Row Exclusive (SX) lock is acquired on table and a DML Exclusive (X) lock is acquired on the partition.
- The DDL timestamp of the table and partition is updated.
- Since a query is executed to check for the existence of rows in the partition that correspond to the literal value being dropped from the partition, it is advisable to create a local prefixed index on the table. This will speed up the execution of the query and the overall operation.
- Status (Index Unusable) of local indexes remains unaffected; however, the index partition being affected retains the new partition-value-list. Global Indexes remain unaffected by this operation.

ALTER TABLE SPLIT PARTITION

You specify the list of values for the first new partition. The second new partition receives the remaining values:

```
SQL> ALTER TABLE locations
  2   SPLIT PARTITION region_east
  3   VALUES( 'CT' , 'VA' , 'MD' )
  4   INTO
  5   (PARTITION region_east_1 TABLESPACE tbs2
  6   ,PARTITION region_east_2 STORAGE (NEXT 2M)
  7   )
  8   PARALLEL 5;
```

ORACLE

8-19

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER TABLE SPLIT PARTITION

The syntax of this statement is modified to support **LIST** partitioned tables. Semantics of this statement are also extended to allow a single partition value list of a partition to be split into two distinct partitions with nonoverlapping value list. The syntax is exactly the same as that for splitting **RANGE** partitions, except that the **AT** keyword is replaced with the **VALUES** keyword.

The list of values following the **VALUES** clause applies to the first partition of the two new partitions being created. The list of values for the second partition is obtained by subtracting the first new partition literal-values list from the original literal-values list of the partition being split.

Additionally the new partitions can have specified partition names, physical attributes, and logging clause.

If the **INTO** clause is not specified, system generated names are used for the two new partitions.

The above example splits the partition **region_east** into two partitions: **region_east_1** with literal-value list of ('CT','VA','MD'), and **region_east_2** inheriting the remaining literal-value list of ('NY','NH','ME','MA','PA','NJ'). The individual partitions have new physical attributes specified at the partition level. This operation is run with a parallelism of degree 5.

ALTER TABLE SPLIT PARTITION (continued)

Usage:

- The set of literal-values specified for the first new partition must be a subset of the set of literal values that describe the current partition value list being split. Oracle will return an error message otherwise.
- No new literal-values can be added to the partition description other than those that existed for the partition being split.
- After the split, rows with partitioning keys with value equal to the specified VALUES list will be inserted into the first partition. Rows with partitioning keys not equal to the specified VALUES list will be inserted into the second partition.
- The new partition will inherit all unspecified physical attributes from the partition being split (not the table-level default).
- This statement also performs a matching split on the corresponding partition in each local index defined on the table. The index partitions are split even if they are marked Index Unusable.
- New local index partitions are assigned the same name as that of the corresponding new base table partition, except for indexes that already have a partition with that name. In such a case, a name will be generated by Oracle with the form SYS_Pn.
- With the exception of TABLESPACE attribute, the physical attributes of the LOCAL index partition being split are used for both new index partitions. If the parent LOCAL index lacks default TABLESPACE attribute, new LOCAL index partitions will reside in the same tablespaces as the corresponding newly created partitions of the underlying table.
- All global indexes will be marked Index Unusable if the partition is not empty.
- ALTER TABLE SPLIT PARTITION locks the table in Row Exclusive (SX) mode and locks partition in Exclusive (X) mode.

List Partitioning Usage

- Only supported for heap tables
- Multi-column partitioning not supported
- The specified literal values must be unique across all literal values of all partition value lists.
- NULL can be specified as a partition literal value.
- MAXVALUE cannot be specified.
- Each list must have at least one literal.
- String values cannot exceed 4K.
- Partition pruning, partition wise joins, and parallelism are supported.
- Local indexes and global range partitioned indexes are supported.

ORACLE

8-21

Copyright © Oracle Corporation, 2001. All rights reserved.

List Partitioning Usage

Generally, all semantics that apply for RANGE method also apply for LIST. Here are some specific features that apply to LIST partition method:

- The LIST method is not extended to index-organized tables in Oracle9i.
- Unlike RANGE and HASH partitioning, multicolumn partitioning is not supported for LIST partitioning. With this method, the partitioning key can consist only of a single column of the table. Otherwise, all columns that can be partitioned by RANGE or HASH can be used.
- The partitions do not have any implicit ordering like RANGE, and thus can be specified in any order.
- The specified literal values that define the partitioning criterion of any given partition have to be unique across all literal values of all partition value lists of the object. If there are duplicate values, Oracle will return an error.
- The keyword NULL can be specified as a value-element of this list, so NULL values for a partitioning key can map to a specific partition. However, one must be cautious with predicates using an IN-LIST clause that involves NULL values, because SQL language semantics treats the NULL literal differently than other literal values. Queries that test equality predicates for NULL literals should be evaluated using existential (is) predicates as opposed to regular equality (=) predicates.

List Partitioning Usage (continued)

- The literal MAXVALUE can no longer be specified as a partition literal value because it has no real meaning.
- The set of literal values that describe a partition value list must have at least one element; that is, it cannot be empty.
- The string comprising the list of values for a partition cannot exceed 4K bytes.

Metadata API

- **It is difficult to get metadata out of an Oracle8i database.**
- **Most popular methods query the data dictionary.**
- **This involves many select statements.**

ORACLE

8-23

Copyright © Oracle Corporation, 2001. All rights reserved.

Metadata API

There are essentially three methods to extract metadata from the data dictionary in Oracle8i:

- The first and most popular method involves querying the data dictionary using SQL statements. This is problematic due to high maintenance costs created by new object definitions and DDL changes. Also, in many cases more than one select statement must be written. This increases network traffic.
- The second method is to run Export with ROWS=N and then to run Import with SHOW=Y. This produces a text file from the binary dump file that can be edited to create SQL scripts. This method can require substantial editing and is not considered a convenient technique.
- The third method involves the OCIDescribeAny interface. This is not widely used due to drawbacks, such as being incapable of retrieving a complete set of metadata about all database objects. It also does not scale well.

Metadata API in Oracle9i

- A new package, **DBMS_METADATA**, is introduced.
- This package allows you to extract metadata in:
 - Browsing mode, or
 - Programmatic mode where the program can specify:
 - The type of objects to be retrieved
 - Various selection criteria
 - Transformation of the output by default is XML, but can be of any format (uses XSL).

ORACLE

8-24

Copyright © Oracle Corporation, 2001. All rights reserved.

Metadata API in Oracle9i

DBMS_METADATA (the Metadata API) package is new for Oracle9i. It allows callers to easily retrieve complete database object definitions (metadata) from the dictionary.

To retrieve database object metadata, callers can specify the following:

- The type of object whose metadata is to be retrieved, for example, tables, indexes, procedures, and so on.
- Selection criteria, such as owner, name, and so on.
- Transformations on the output. By default, the output is represented in XML. However, callers can specify transformations, which are implemented by XSL-T style sheets that can be stored in the database or externally. In particular, callers can ask that the metadata be transformed into SQL DDL.

DBMS_METADATA provides two types of retrieval interfaces for two anticipated types of usage:

- For programmatic use: OPEN, SET_FILTER, SET_COUNT, GET_QUERY, SET_PARSE_ITEM, ADD_TRANSFORM, SET_TRANSFORM_PARAM, FETCH_XXX, and CLOSE. These allow flexible selection criteria and the extraction of a stream of objects.
- For browsing: GET_XML and GET_DDL return metadata for a single object. These are designed for use in SQL queries and for ad hoc browsing.

Metadata API in Oracle9i Browsing Example

```
SQL> SELECT dbms_metadata.get_ddl  
          ( 'TABLE' , 'SALES' )  
2  FROM    dual;
```

- The query can have a **WHERE** clause.
- Output is in SQL but can be in XML format.
- Combine with a **SPOOL** statement for XML file.
- Any combination of objects can be extracted depending on the result of the select statement.
- Running spool while extracting leads to a file that can be edited immediately.

ORACLE

8-25

Copyright © Oracle Corporation, 2001. All rights reserved.

Metadata Unload (continued)

To address the metadata unload problem, Oracle9i has a new option. This form of extraction offers many advantages over using EXPORT. Using export gives the user the following options: entire database, users schema, or table. To extract many tables it is necessary to perform multiple export commands, each exporting out a single object.

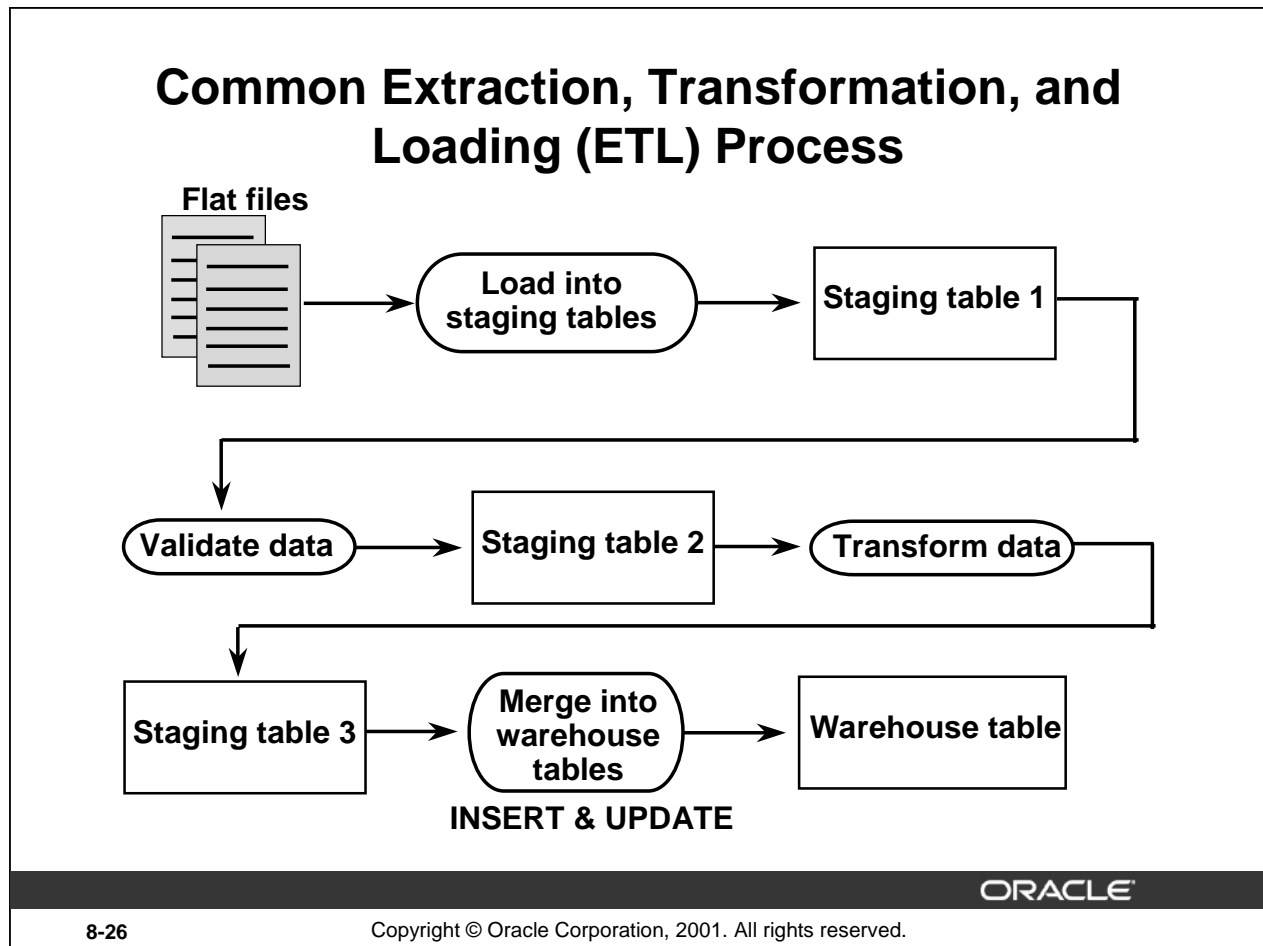
The above SELECT statement provides, in combination with a SPOOL statement, an editable output of the required objects. In the above example, it is assumed that when executing this query, you are connected as SH.

You can also use another version of the browsing function: GET_XML which outputs XML format.

These functions provide a simple way to return the metadata for a single object. Conceptually each GET_XXX call is comprised of an OPEN, one or two SET_FILTER calls, optionally an ADD_TRANSFORM, a FETCH_XXX and a CLOSE.

These functions can only be used to fetch named objects. For the others, use the programmatic interface.

For all syntax information refer to the *Oracle9i Supplied PL/SQL Packages Reference Guide*.



Common Extraction, Transformation, and Loading (ETL) Process

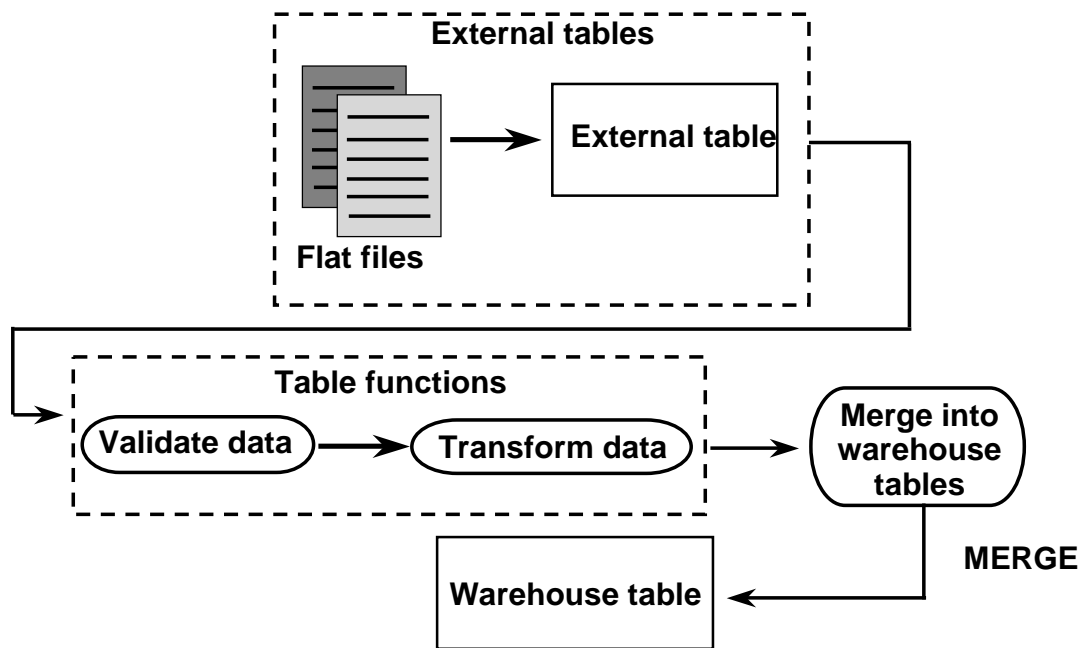
Data transformations are often the most complex and, in terms of processing time, the most costly part of the Extraction Transformation Loading (ETL) process. They can run the gamut from simple data conversions to extremely complex data scrubbing techniques. Many, if not all, data transformations can occur within an Oracle9i database, although transformations are often implemented outside of the database (for example, on flat files) as well.

From an architectural perspective, you can transform your data in two ways:

- Multi-Stage Data Transformation
- Pipelined Data Transformation

With Multi-Stage Data Transformation, the data transformation logic for most data warehouses consists of multiple steps. For example, in transforming new records to be inserted into a sales table, there may be separate logical transformation steps to validate each dimension key. A graphical way of looking at the transformation logic is presented on the above slide. When using Oracle8i as a transformation engine, a common strategy is to implement each different transformation as a separate SQL operation and to create a separate, temporary staging table to store the incremental results for each step. This load-then-transform strategy also provides a natural checkpointing scheme to the entire transformation process, which enables to the process to be more easily monitored and restarted. However, a disadvantage is that, due to the multistaging, the required space and time increases.

Pipelined Data Transformation in Oracle9i



ORACLE

8-27

Copyright © Oracle Corporation, 2001. All rights reserved.

Pipelined Data Transformation in Oracle9i

With the introduction of Oracle9i, Oracle's database capabilities have been significantly enhanced to address specifically some of the tasks in ETL environments. The ETL process flow can be changed dramatically, and the database becomes an integral part of the ETL solution. Taking advantage of the new functionality, some of the former necessary process steps become obsolete while some others can be remodeled to enhance the data flow and the data transformation to become more scalable and noninterruptive. We are no longer talking about serial transform-then-load (with most of the tasks done outside the database) or load-then-transform; rather we are talking about an enhanced transform-while-loading.

Oracle9i offers a wide variety of new capabilities to address all the issues and tasks relevant in an ETL scenario. It is important to understand that the database offers toolkit functionality rather than trying to address a one-size-fits-all solution. The underlying database has to enable the most appropriate ETL process flow for a specific customer need, and not dictate or constrain it from a technical perspective. The above slide illustrates the new functionality, which is discussed throughout later slides in this lesson.

Overview of External Tables

- External tables are *read-only* tables where the data is stored outside the database in flat files.
- The data can be queried like a *virtual table*, using any supported language inside the database.
- No indexes can be created.
- The metadata for an external table is created using a `CREATE TABLE` statement.
- Access rights are controlled through `SELECT TABLE` and `READ/WRITE DIRECTORY` privileges.
- An external table's definition describes how the external data should be presented to the database.

ORACLE

8-28

Copyright © Oracle Corporation, 2001. All rights reserved.

Overview of External Tables

External tables are like regular SQL tables with the exception that the data is read only and does not reside in the database, thus the organization is external. The external table can be queried directly and in parallel using SQL. As a result, the external table acts as a view. The metadata for the external table is created using the "CREATE TABLE ... ORGANIZATION EXTERNAL" statement.

No DML operations are possible and no indexes can be created on them.

The CREATE TABLE ... ORGANIZATION EXTERNAL operation involves only the creation of metadata in the Oracle Dictionary since the external data already exists outside the database. Once the metadata is created, the external table feature enables the user to easily perform parallel extraction of data from the specified external sources.

Applications of External Tables

- **Allow external data to be queried and joined directly and in parallel, without requiring it to be loaded into the database.**
- **Eliminate the need for staging the data within the database for ETL in data warehousing applications.**
- **Are useful in environments where an external source must be joined with database objects and then transformed.**
- **Are useful when the external data is large and not queried frequently.**
- **Complement SQL*Loader functionality.**

ORACLE

8-29

Copyright © Oracle Corporation, 2001. All rights reserved.

Applications of External Tables

Oracle9i's external table feature allows you to use external data as a *virtual table* and can be queried and joined directly and in parallel without requiring the external data to be first loaded in the database.

External tables enable the pipelining of the loading phase with the transformation phase. The transformation process can be merged with the loading process without any interruption of the data streaming. It is no longer necessary to stage the data inside the database for comparison or transformation.

The main difference between external tables and regular tables is that externally organized tables are read-only. No DML operations are possible and no indexes can be created on them. Oracle9i's external tables are a complement to the existing SQL*Loader functionality, and are especially useful for environments where the complete external source has to be joined with existing database objects and transformed in a complex manner or the external data volume is large and used only once.

SQL*Loader, on the other hand, might still be the better choice for loading of data where additional indexing of the staging table is necessary. This is true for operations where the data is used in independent complex transformations or the data is only partially used in further processing.

Example of Defining External Tables

```
CREATE table employees_ext (employee_id NUMBER,  
first_name CHAR(30), last_name CHAR(30), ...)  
ORGANIZATION EXTERNAL (      -- External Table  
TYPE oracle_loader           -- Access Driver  
DEFAULT DIRECTORY delta_dir  -- Files Directory  
ACCESS PARAMETERS           -- Similar to SQL*Loader  
(RECORDS DELIMITED BY NEWLINE  
  FIELDS TERMINATED BY ', '  
  BADFILE 'bad_emp_ext'  
  LOGFILE 'log_emp_ext'  
  MISSING FIELDS ARE NULL)  
  LOCATION ('emp1.txt','emp2.txt'))  
PARALLEL 5      -- Independent from the number of files  
REJECT LIMIT UNLIMITED;
```

ORACLE

8-30

Copyright © Oracle Corporation, 2001. All rights reserved.

Example of Defining External Tables

An external table can be created with a single `CREATE TABLE` command. This creates the meta information which is necessary to access the external data seamlessly from inside the database.

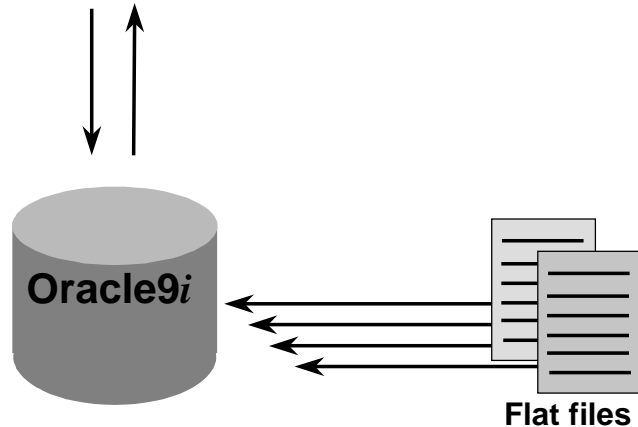
The following information must be provided:

- Columns and data types for access in the database
- Where to find the external data
- Access driver: It is the responsibility of the access driver and the external table layer to do the necessary transformations required on the data in the data file so that it matches the external table definition. There is one access driver for every implementation of an external table type. Right now, `ORACLE_LOADER` is the only access driver available.
- Format of the external data, similar to `SQL*Loader`
- Degree of parallelism: The degree is not dependent on the number of external data files.

In the example above, an external table named `employees_ext` is defined. `DELTA_DIR` is a directory where the external flat files are residing. This example also use the default access driver for this implementation of external tables. It is called `oracle_loader`. The access parameters control the extraction of data from the flat file using record and file formatting information. The directory object was introduced in Oracle8i.

Querying External Tables

```
SQL> SELECT * FROM employees_ext;
```



ORACLE

8-31

Copyright © Oracle Corporation, 2001. All rights reserved.

Querying External Tables

In the above example, when the external table `employees_ext` is queried, the data is retrieved from the external data files. When the user selects data from the external table, the dataflow goes from the external data source to the Oracle SQL engine where data is processed. As data is extracted, it is transparently converted by the access driver from its external representation into an equivalent Oracle native representation (the loader stream).

Data Dictionary Information for External Tables

- **DBA_EXTERNAL_TABLES**
 - OWNER
 - NAME
 - TYPE_OWNER
 - TYPE_NAME
 - DEFAULT_DIRECTORY_OWNER
 - DEFAULT_DIRECTORY_NAME
 - REJECT_LIMIT
- **DBA_EXTERNAL_LOCATIONS**
 - OWNER
 - TABLE_NAME
 - LOCATION
 - DIRECTORY_OWNER
 - DIRECTORY_NAME

ORACLE

8-32

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Dictionary Information for External Tables

DBA_EXTERNAL_TABLES lists the specific attributes of all the external tables in the system:

- OWNER: Owner of the external table
- NAME: Name of the external table
- TYPE_OWNER: Implementation type owner
- TYPE_NAME: Implementation type name
- DEFAULT_DIRECTORY_OWNER: Owner of the default directory for this external table
- DEFAULT_DIRECTORY_NAME: Name of the default directory for this external table
- REJECT_LIMIT: Reject limit

DBA_EXTERNAL_LOCATIONS lists the specific flat files and corresponding Oracle Directories

- OWNER: Owner of the external table
- TABLE_NAME: Name of the external table
- LOCATION: Flat file name
- DIRECTORY_OWNER: Owner of the directory for this external table
- DIRECTORY_NAME: Name of the directory for this external table

Summary

In this lesson, you should have learned how to:

- **Use the new `UPDATE GLOBAL INDEXES` clause**
- **Create `ORGANIZATION EXTERNAL` tables**
- **Create partitioned tables with the `LIST` method**
- **Use the `DBMS_METADATA` package to unload meta data**

ORACLE

Practice 8-1 Overview

This practice covers the following topics:

- Create an Oracle Directory
- Create an external table
- Use the new `EXTERNAL_TABLE SQL*Loader` parameter

ORACLE

Practice 8-2 Overview

This practice covers the following topics:

- Create a LIST-partitioned table
- Show how partition pruning is done with a LIST-partitioned table
- Use the `DBMS_METADATA` package in browsing mode

ORACLE

9

Segment Management (Part II)

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Understand and use the new Automatic Segment-Space Management functionality**
- **Create and use bitmap join indexes**

ORACLE

Automatic Segment-Space Management

- **Automatic Segment-Space Management introduces the capability of tracking in-segment free and used space through bitmaps as opposed to the previously used free lists.**
- **The benefits provided by this new capability include:**
 - **Ease of use**
 - **Better space utilization**
 - **Better concurrency handling**
 - **Better performance**

ORACLE

9-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Automatic Segment-Space Management

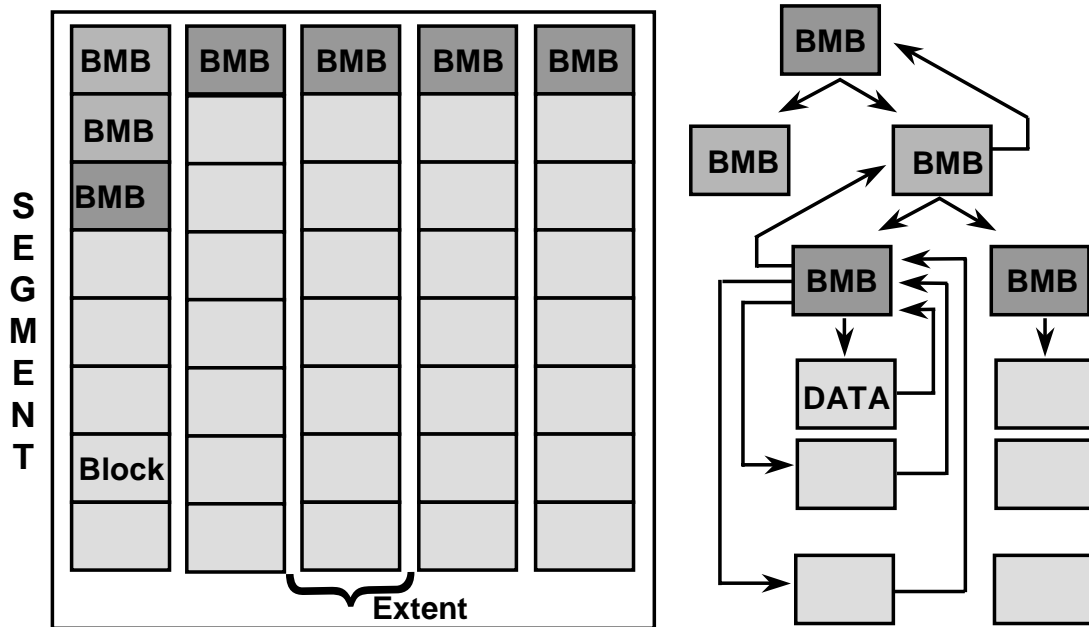
Automatic Segment-Space Management introduces a new scheme of managing free space inside the database segments. It provides the capability to track in-segment free and used space through bitmaps as opposed to the previously used free lists. With Automatic Space Management segments, a bitmap describes a status of each block in the segment with respect to its fullness, to a certain degree of precision. The map is contained in a separate state of blocks. When a new row is inserted, the map is searched for a block with sufficient space. When the block becomes more full or less full, its new state is reflected in the bitmap.

This new implementation provides the following benefits:

- **Ease of Use:** Achieved by requiring fewer space-related options that are often needed and occasionally misused (PCTUSED, FREELISTS, FREELIST GROUPS).
- **Better Space Utilization:** Especially for the objects with highly varying size rows.
- **Better Concurrency Handling:** Better run-time adjustment to variations in concurrent access
- **Better Performance:** Improved multi-instance behavior in terms of performance/space utilization

Note: One can only create such segments in a locally managed tablespace. Thus, segments stored in the SYSTEM tablespace cannot be of that kind.

Automatic Segment-Space Management at Work



ORACLE

9-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Automatic Segment-Space Management at Work

Automatic Space Management segments have a set of bitmap blocks (BMBs) describing the space utilization of the data blocks in that segment. For each data block, there is a set of bits per block that gives an idea of space available in that block.

BMBs are organized in a tree hierarchy. The maximum number of levels inside this hierarchy is three. The leaves of this hierarchy represent the space information for a set of contiguous data blocks that belong to the segment. The BMB leaves are the unit at which space has affinity to an instance in a multiinstance environment.

When segments are growing with larger amounts of data being stored in the database, it becomes prohibitively expensive to search all the leaves of that hierarchy during an insert. Thus, an intermediate level of BMBs is created to contain meta-data about the leaves.

The root level of the hierarchy, which stores the references to all intermediate BMBs, is stored in what was called the segment header.

During an INSERT statement, starting from the root of the hierarchy, Oracle grabs the segment header from where it will be able to go down to the leaves to find a useful Bit Map Block, which points to data blocks containing free space. These BMBs are acquired in shared mode so that multiple processes can search into the same BMB.

Automatic Segment-Space Management at Work (continued)

Within an intermediate BMB, concurrent processes get distributed by hashing on the <instance number, process id> . To simplify, this operation will get BMB leaves. Once again, Oracle uses hashing on the process ID to provide a starting point again inside this BMB leaf.

In this kind of BMB's hierarchy, the leaves point to a range of data blocks that are potential candidates for the insert operation.

Creating an Automatic Space Management Segment

- These segments are declared at the tablespace level.
- The tablespace must be permanent and locally managed.
- **SEGMENT SPACE MANAGEMENT** is the attribute used for tablespace creation, which cannot be subsequently altered:
 - Automatic Space Management segments are specified through the **AUTO** keyword.
 - For freelist segments, use the default value of **MANUAL**.
- **Specifications of PCTUSED, FREELIST, and FREELIST GROUPS are ignored at table creation.**

ORACLE

Creating an Automatic Space Management Segment: Example

```
CREATE TABLESPACE sample
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;

CREATE TABLE employees(...)
TABLESPACE sample;
```

ORACLE

9-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Creating an Automatic Space Management Segment: Example

The SEGMENT SPACE MANAGEMENT specification only applies to the level of a tablespace. This means that if you want an automatically free-space managed table, it needs to be created in a tablespace like the one exposed in the above example. Thus, it will not be possible to specify the SEGMENT SPACE MANAGEMENT clause at the table level.

The justifications of what seems to be a restriction are in fact only syntax restrictions for the following reasons:

- It is useful to allow it on the tablespace level, so that specification of an individual object creation does not have to include it.
- This avoids adding even more syntax to the creation of database objects, which is already difficult enough.
- Tablespace reorganization will be made easier if all segments in the tablespace adhere to the same management policy (this is only planned for future releases).
- Justification for limiting it to the LOCALLY MANAGED tablespaces is similar: tablespace reorganization will be easier if the tablespace is bitmapped.

Note: Segments that can be automatically free-space managed are heap tables, indexes, IOTs, and LOBs. Note also that the above example assumes the use of Oracle Managed Files (see later in this course).

Modifications to the DBMS_SPACE Package

- The **FREE_BLOCKS** procedure returns an error if called on a bitmapped segment.
- The new **SPACE_USAGE** procedure provides information about free blocks in Automatic Space Management segments:

- | | |
|-----------------------------|---------------------|
| — SEGMENT_OWNER | — FS1_BLOCKS |
| — SEGMENT_NAME | — FS2_BLOCKS |
| — PARTITION_NAME | — FS3_BLOCKS |
| — SEGMENT_TYPE | — FS4_BLOCKS |
| — UNFORMATTED_BLOCKS | |

ORACLE

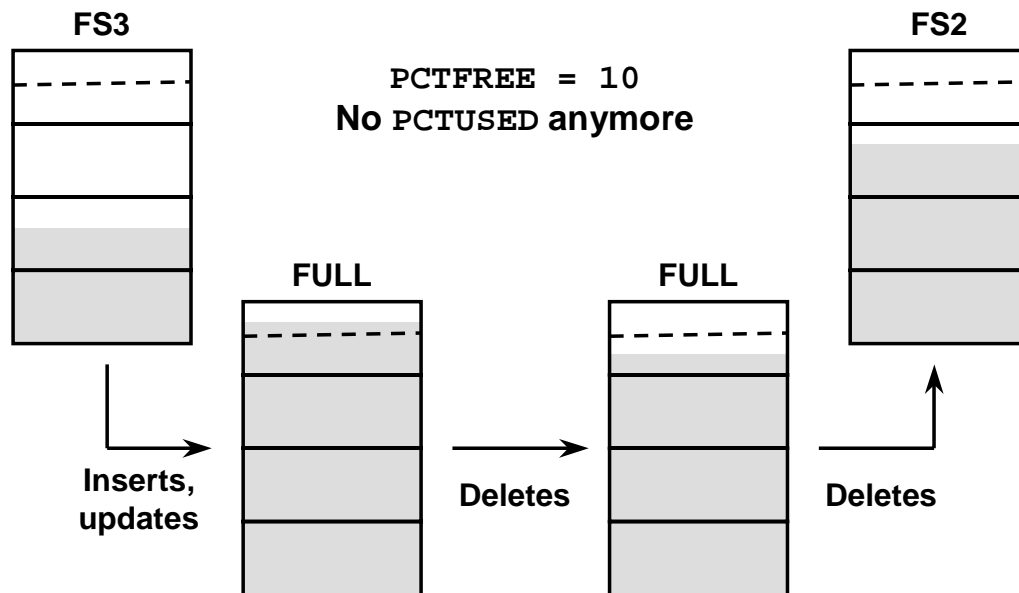
Modifications to the DBMS_SPACE Package

SPACE_USAGE is a new procedure that returns information about free blocks below the HWM in the Automatic Space Management segment, and in this sense it is similar to the **FREE_BLOCKS** procedure for the freelist segments. This procedure has the following input parameters (not all parameters are listed here):

- **SEGMENT_OWNER**: Schema name of the segment
- **SEGMENT_NAME**: Segment name
- **PARTITION_NAME**: (optional) Name of a partition (NULL for non-partitioned objects)
- **SEGMENT_TYPE**: Type of the segment (for example: TABLE, INDEX)
- **INSTANCE_NUMBER**: (optional) Instance number of instance for which information is returned. NULL returns everything
- **UNFORMATTED_BLOCKS**: Total number of blocks that are unformatted
- **FS1_BLOCKS**: OUT Blocks below HWM with between 0 to 25% free space
- **FS2_BLOCKS**: OUT Blocks below HWM with between 25 to 50% free space
- **FS3_BLOCKS**: OUT Blocks below HWM with between 50 to 75% free space
- **FS4_BLOCKS**: OUT Blocks below HWM with between 75 to 100% free space
- **FULL_BLOCKS**: OUT Blocks which are no longer available for INSERTs.

Note: If you invoke **FREE_BLOCKS** procedure on an Automatic Space Management segment, you will get an ORA-10618: Operation not allowed on this segment

Block Space Management



ORACLE

9-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Block Space Management

Automatic Segment-Space Management does not use PCTUSED markers; each block is divided into four sections, named FS1, FS2, FS3, and FS4.

A particular block is considered as being an:

- FS1 block if it contains between 0 and 25% of free space
- FS2 block if it contains between 25% and 50% of free space
- FS3 block if it contains between 50% and 75% of free space
- FS4 block if it contains between 75% and 100% of free space

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, it is possible to tell if 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.

On the above 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 block.

The block is considered for insertion again, as soon as its free space level drops below the *next* section; therefore, the block is still considered FULL in the third situation above. In the above case, it gets status FS2 as soon as the free space is more than 25%.

DBMS_REPAIR Package

- **Support for Automatic Segment-Space Management**
- **Use the `SEGMENT_FIX_STATUS` procedure to update free space status bits. There are two options:**
 - **Recalculate the status for all blocks of a segment, based on current contents.**
 - **Set the status to a specific value for a specific block (for example, in case of corruption)**

ORACLE

9-10

Copyright © Oracle Corporation, 2001. All rights reserved.

SEGMENT_FIX_STATUS Procedure

`SEGMENT_FIX_STATUS` is a new procedure that allows you to manually fix the bitmap status of data blocks. Here is the list of its input parameters:

`SEGMENT_OWNER`: Schema name of the segment interested in

`SEGMENT_NAME`: Segment name

`PARTITION_NAME`: (optional) Name of an individual partition (NULL for non-partitioned objects)

`SEGMENT_TYPE`: Type of the segment (for example: TABLE, INDEX)

`FILE_NUMBER`: Tablespace relative file number of the block whose status has to be fixed

`BLOCK_NUMBER`: File relative file number of the block whose status has to be fixed

`STATUS_VALUE`: (optional) Value to which the block status described by the `FILE_NUMBER` and `BLOCK_NUMBER` will be set. If omitted, the status will be set, based on the block's current state. This should almost always be the case, but if there is a bug in the calculation algorithm, the value can be set manually.

Note: Always contact Oracle Support Services before invoking this procedure with the `status_value` parameter.

DBMS_REPAIR and PCTFREE Implementation

- With FREELIST segments, when you want to change the PCTFREE value, use an ALTER command.
- With Automatic Space Management segments, there is *no* status recomputation at ALTER time:
 - ALTER command *and*
 - Invoke DBMS_REPAIR.SEGMENT_FIX_STATUS procedure to rebuild the bitmap status for data blocks

```
SQL> ALTER TABLE employees PCTFREE 40;
```

```
SQL> execute dbms_repair.segment_fix_status( -  
      2  'HR' , 'EMPLOYEES' );
```

ORACLE

9-11

Copyright © Oracle Corporation, 2001. All rights reserved.

DBMS_REPAIR and PCTFREE Implementation

Once the PCTFREE is changed, the current BMBs status for some of the existing blocks may become incorrect, and at some point the status must be re-computed. It is difficult to re-compute a PCTFREE at ALTER time for Automatic Space Management segments as a Full Table Scan will be required, which would slow the ALTER command significantly. Therefore, status (in this case, the bitmap blocks) will *not* be re-computed at ALTER time, but during a manual execution of the DBMS_REPAIR.SEGMENT_FIX_STATUS procedure.

Note: ANALYZE VALIDATE STRUCTURE validates the bitmap information for the Automatic Space Management segment.

Modifications to Dictionary Views

- **DBA_TABLESPACES and USER_TABLESPACES:**
New column SEGMENT_SPACE_MANAGEMENT
 - **MANUAL:** Space management with free lists
 - **AUTO:** Space management with bitmaps
- **DBA_TABLES: Changed column meanings (only for automatic space management tables)**
 - **BLOCKS:** Number of blocks read by a full table scan
 - **EMPTY_BLOCKS:** Remainder of blocks not read by the full table scan
 - **PCTUSED, FREELISTS, and FREELIST_GROUPS:** values are NULL

ORACLE

9-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Modifications to Dictionary Views

DBA_TABLESPACES: The new column `SEGMENT_SPACE_MANAGEMENT` added to this view, indicates whether or not the segment management in this tablespace is managed using free lists (`MANUAL`) or bitmaps (`AUTO`)

DBA_TABLES: With freelist segments this view has a column called `BLOCKS`, which contains the number of formatted blocks, or blocks below High Water Mark. There is also a column called `EMPTY_BLOCKS`, which contains the number of blocks above High Water Mark (unformatted blocks).

For Automatic Space Management segments, formatted blocks and blocks below the high water mark are not always the same; therefore, the `BLOCKS` value will represent the number of blocks read by the full table scan, and the remainder blocks not read by the full table scan are listed as `EMPTY_BLOCKS`.

Note: The `DBMS_SPACE.SPACE_USAGE` procedure provides some information for `EMPTY_BLOCKS` as well.

Compatibility and Migration

- **Automatic Segment-Space Management is only enabled if the COMPATIBLE parameter is set to 9.0.0 or higher.**
- **In order to downgrade to a pre Oracle9i release, tablespaces with Automatic Space Management segments must be dropped.**
- **Identify tablespaces with Automatic Space Management segments:**

```
SQL> SELECT tablespace_name
2  FROM    dba_tablespaces
3  WHERE   segment_space_management = 'AUTO';
```

ORACLE

Compatibility and Migration

When downgrading, you must move any existing objects to a tablespace with freelist segment management, since no explicit migration is being provided from the bitmapped format to the freelist format.

Compatibility and Migration

- **There is no explicit procedure to convert a `FREELIST` segment to an Automatic Space Management segment.**
- **In order to migrate existing `FREELIST` segments to Automatic Space Management segments, you need to:**
 - **Create tablespaces with Automatic Segment-Space Management feature**
 - **Move the objects to new tablespaces**

ORACLE

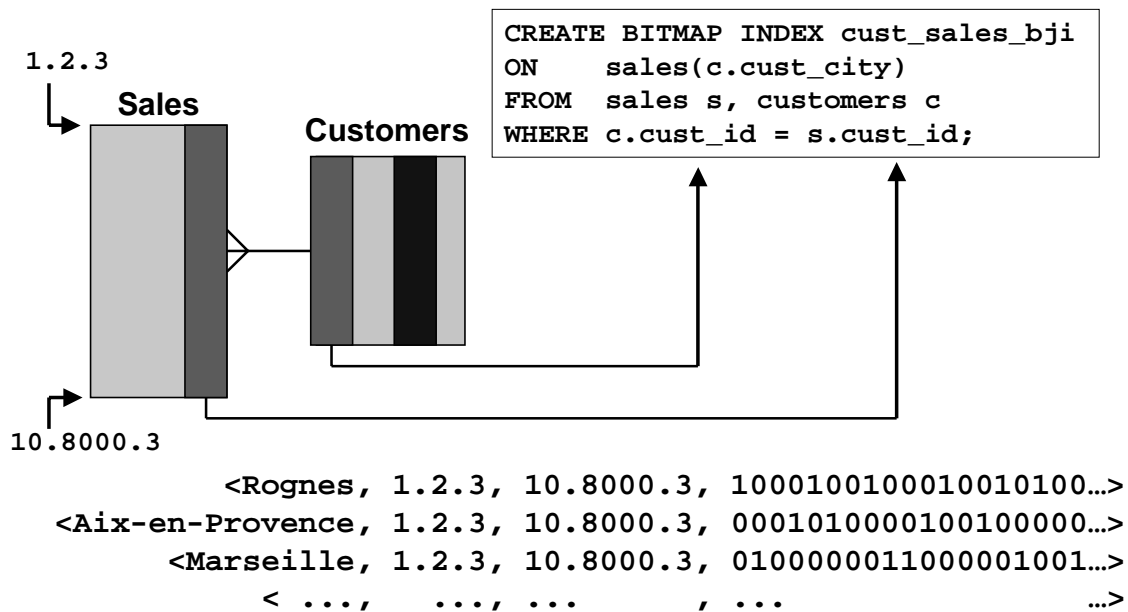
9-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Compatibility and Migration (continued)

In order to convert existing data, the user must create tablespaces with Automatic Segment-Space Management and move the objects into them. Existing mechanisms, such as `ALTER ... MOVE`, and `CREATE ... AS SELECT` can be used for this purpose. They will work automatically, because the segments will inherit Automatic Space Management from the tablespace in which they are created.

What Is a Bitmap Join Index?



ORACLE

9-15

Copyright © Oracle Corporation, 2001. All rights reserved.

What Is a Bitmap Join Index?

In addition to a bitmap index on a single table, you can create a bitmap join index in Oracle9i. A bitmap join index is a bitmap index for the join of two or more tables. A bitmap join index is a space efficient way of reducing the volume of data that must be joined by performing restrictions in advance.

As you can see on the above slide, Oracle9i introduces a new CREATE BITMAP INDEX syntax allowing you to specify a FROM and a WHERE clause.

Here, we create a new bitmap join index named cust_sales_bji on the SALES table. The key of this index is the column CUST_CITY of the CUSTOMERS table. This example assumes that there is an enforced primary key constraint on CUSTOMERS in order to ensure that what is stored in the bitmap reflect the reality of the data in the tables. The column CUST_ID is the primary key of CUSTOMERS but is also a foreign key inside SALES (although not required).

The FROM and WHERE clause in the CREATE statement allow Oracle9i to make the link between the two tables. They represent the natural join condition between the two tables.

The middle part of the above graphic shows you a theoretical implementation of this bitmap join index. Each entry or key in the index represents a possible city found in the CUSTOMERS table. A bitmap is then associated to one particular key. The meaning of the bitmap is quite obvious as it has the same representation as for traditional bitmap indexes. Each bit in a bitmap corresponds to one row in the SALES table. In the first key above (Rognes), you can see that the first row in the SALES table corresponds to a product sold to a Rognes customer, while the second bit is not a product sold to a Rognes customer.

Oracle9i: New Features for Administrators 9-15

Advantages and Disadvantages

Bitmap join indexes provide an indexing structure across two or more tables.

- **Advantages:**

- **Good join query performance**
- **Space efficiency**
- **Useful for large dimension tables in star schemas**

- **Disadvantages:**

- **More indexes are required: up to one index per dimension table column rather than one index per dimension table.**
- **Maintenance costs are higher: building or refreshing a bitmap join index requires a join.**

ORACLE

9-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Advantages and Disadvantages

A Bitmap join index is an index on one table that involves columns of one or more different tables through a join.

The volume of data that must be joined can be reduced if bitmap join indexes are used as joins have already been precalculated. In addition, bitmap join indexes, which can contain one or multiple dimension tables, can eliminate bit wise operations that are necessary in the star transformation's use of bitmap indexes.

An alternative to a bitmap join index is a materialized join view which is the materialization of a join in a table. Compared to a materialized join view, a bitmap join index is much more space efficient as it compresses `rowids` of the fact tables.

Also, queries using bitmap join indexes can be sped up through bit wise operations.

On the other hand, you may need to create more bitmap join indexes on the fact table to satisfy the maximum number of different queries. This means that you may have to create one bitmap join index for each column of the corresponding dimension table(s). Of course, having many indexes on one table results in higher maintenance costs, especially when the fact table is updated.

Example With Three Tables



```
SQL> CREATE BITMAP INDEX c_s_p_bji ON
2   sales(c.cust_gender,p.prod_category)
3   FROM   sales s, customers c, products p
4   WHERE  c.cust_id = s.cust_id
5   AND    p.prod_id = s.prod_id;
```

```
SQL> SELECT SUM(s.amount_sold)
2   FROM   sales s
3   ,      customers c
4   ,      products p
5  WHERE   s.cust_id = c.cust_id
6  AND     s.prod_id = p.prod_id
7  AND     c.cust_gender = 'MALE'
8  AND     p.prod_category = 'MOBILE';
```

ORACLE

9-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Example With Three Tables

You can create a bitmap join index on more than one table, as in the above slide, where `c_s_p_bji` is a bitmap join index on the `SALES` table that indexes two columns in two different tables: `CUST_GENDER` in `CUSTOMERS` and `PROD_CATEGORY` in `PRODUCTS`.

The above query that retrieves the total `MOBILE` sales for `MALE` customers can use the `c_s_p_bji` index.

Note: The above example clarifies the syntax possibilities. In the above case, you might also consider to create two different bitmap join indexes instead of just one, because this could cause an explosion in the number of indexes (one for each query); whereas, creating one bitmap join index on the `SALES` table for `CUST_GENDER` and one bitmap join index on `SALES` for `PROD_CATEGORY` allows the optimizer to use both indexes for the above query. This is less efficient than the above solution with the single index, but has the added advantage of allowing queries only between `SALES` and `CUSTOMERS`, or `SALES` and `PRODUCTS`, for example.

Data Dictionary and Bitmap Join Indexes

- **DBA_INDEXES** contains a new column:
JOIN_INDEX: TRUE/FALSE
- **DBA_IND_COLUMNS** points to the corresponding dimension table.
- **DBA_JOIN_IND_COLUMNS** is a brand new view giving you the join conditions used to create the index.

ORACLE

9-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Data Dictionary and Bitmap Join Indexes

The *_INDEXES views contain a brand new column called JOIN_INDEX which value is TRUE if the corresponding index is a Bitmap Join Index, and FALSE otherwise.

Inside *_IND_COLUMNS views, for bitmap join indexes, the TABLE_NAME and TABLE_OWNER columns in this view may not match the TABLE_NAME and TABLE_OWNER columns you find in the *_INDEXES (and other similar) data dictionary views. This indicates the corresponding dimensions tables used by the Bitmap Join Index.

*_JOIN_IND_COLUMNS views describe the join conditions of bitmap join indexes to which you have access. Bitmap join indexes are indexes built on a child table with an index key containing columns from associated parent tables, where all of the tables are connected through join conditions. There is one row for each join condition.

Bitmap Join Indexes Restrictions

- Only one table can be updated concurrently.
- No table can appear twice in the join.
- Cannot be created on index-organized tables or temporary tables
- The dimension table join columns must either:
 - be primary key columns,
 - have unique constraints
- If a dimension table has composite primary key, each column in the primary key must be part of the join.

ORACLE

9-19

Copyright © Oracle Corporation, 2001. All rights reserved.

Bitmap Join Indexes Restrictions

Due to the necessity of storing the results of join results, bitmap join indexes have the following restrictions:

- Only one table can be updated concurrently by different transactions when using the bitmap join index.
- No table can appear twice in the join.
- You cannot create a bitmap join index on an index-organized table or a temporary table.
- The dimension table join columns must be either primary key columns or have unique constraints.
- If a dimension table has composite primary key, each column in the primary key must be part of the join.

Summary

In this lesson, you should have learned how to:

- **Use Automatic Space Management segments**
- **Create bitmap join indexes**

ORACLE

Practice 9-1 Overview

This practice covers the following topics:

- **Monitor space usage in a segment with Automatic Space Management**
- **Create a tablespace with `SEGMENT SPACE MANAGEMENT` set to `AUTO`**
- **Compare free space management with a freelist segment**

ORACLE

10

Performance Improvements

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Use the indexes monitoring feature**
- **Define a skip scan index access**
- **Describe the cursor sharing enhancements**
- **Identify cached execution plans**
- **Use the new first rows optimization**
- **Gather system statistics**

ORACLE

Identifying Unused Indexes

- **Perform monitoring at query parse time.**
- **Turn monitoring on or off during light system activity.**
- **Main benefits include:**
 - **Space conservation**
 - **Improved performance by eliminating unnecessary overhead during DML operations**

ORACLE

10-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Identifying Unused Indexes

Oracle provides a means of monitoring indexes to determine if they are being used or not used. If it is determined that an index is not being used, then it can be dropped, thus eliminating unnecessary statement overhead.

Index Usage Statistics is obtained at query parse time. At parse time we know which indexes should be accessed by a particular query.

When users turn on or off the monitor, Oracle invalidates and recompiles those already compiled cursors dependent on this index in order to re-collect or stop collecting statistics of which indexes are accessed. It would cost some time to activate the monitor in the middle of application execution. Therefore, it is suggested to turn on or off this feature before starting or after stopping the application, or in periods of light activity.

Note: Because Oracle9i collects the statistics at parse time, if a SQL statement is already being executed when the monitoring feature is activated, the indexes accessed by this statement are not counted.

Enabling and Disabling Monitoring Index Usage

- **Start monitoring the usage of an index:**

```
SQL> ALTER INDEX EMPLOYEES_idx  
2 MONITORING USAGE;
```

- **Stop monitoring the usage of an index:**

```
SQL> ALTER INDEX EMPLOYEES_idx  
2 NOMONITORING USAGE;
```

ORACLE

10-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Enabling and Disabling Monitoring Index Usage

Use this clause to begin or end the collection of usage statistics on an index. This clause determines whether an index is being used.

Specify `MONITORING USAGE` to begin statistics collection. Oracle first clears existing statistics on index and then begins to collect statistics on index usage.

To terminate collection of statistics on index, specify `NOMONITORING USAGE`.

V\$OBJECT_USAGE View

- **Displays information about index usage:**
 - **Monitoring on (Y/N)**
 - **Index is used (Y/N)**
 - **Start and stop monitoring timeframe**
- **Each time the MONITORING USAGE clause is specified, the view is reset for the specified index.**
- **The view is never purged unless the index is dropped.**

ORACLE

10-5

Copyright © Oracle Corporation, 2001. All rights reserved.

New View to Support Identifying Unused Indexes

The view V\$OBJECT_USAGE can be queried for the index being monitored to see if the index has been used. The view contains a USED column whose value is YES or NO, depending upon if the index has been used within the time period being monitored. The view also contains the start and stop times of the monitoring period, and a MONITORING column (YES/NO) to indicate if usage monitoring is currently active.

Each time that you specify MONITORING USAGE, the V\$OBJECT_USAGE view is reset for the specified index. The previous usage information is cleared or reset, and a new start time is recorded. When you specify NOMONITORING USAGE, no further monitoring is performed, and the end time is recorded for the monitoring period.

Until the next ALTER INDEX ... MONITORING USAGE statement is issued, the view information is left unchanged unless you drop the corresponding index.

Note: The V\$OBJECT_USAGE is based on a real data dictionary table. This means that its contents are consistent even after an instance crash. Also because the view is dynamic, the contents are dependent on the user querying the view.

New View to Support Identifying Unused Indexes (continued)

V\$OBJECT_USAGE:

- INDEX_NAME: The index name
- TABLE_NAME: The corresponding table
- MONITORING: Indicates whether monitoring is ON or OFF
- USED: Indicates YES or NO whether index has been used during the monitoring time
- START_MONITORING: Time monitoring began on index.
- END_MONITORING: Time monitoring stopped on index.

Skip Scanning of Indexes

- **Use a composite index when the leading column value is unknown.**
- **Supported for:**
 - **Cluster indexes**
 - **Descending index scans**
 - **CONNECT BY clauses**
- **Not supported for reverse key, bitmap indexes, domain indexes, and function-based indexes**

ORACLE

10-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Skip Scanning Definition

In prior releases, a composite index would only be used if the index prefix (leading) column was included in the predicate of the statement. With Oracle9i, the optimizer can use a composite index even if the prefix column value is not known. The optimizer uses an algorithm called skip scanning to retrieve ROWIDs for values that do not use the prefix column. Skip scans reduce the need to add an index to support occasional queries which do not reference the prefix column of an existing index. This can be useful when high levels of DML activity would be degraded by the existence of too many indexes used to support infrequent queries. The algorithm is also valuable in cases where there are no clear advantages as to which column to use as the prefix column in a composite index. The prefix column should be the most discriminating, but also the most frequently referenced in queries. Sometimes, these two requirements are met by two different columns in a composite index, forcing a compromise or the use of multiple indexes.

During a skip scan, the B*-tree is probed for each distinct value in the prefix column. Under each prefix column value, the normal search algorithm takes over. The result is a series of searches through subsets of the index, each of which appears to result from a query using a specific value of the prefix column. However, with the skip scan, the value of the prefix column in each subset is obtained from the initial index probe rather than from the command predicate.

In addition to standard B*-tree indexes, the optimizer can use skip scans for processing cluster indexes, descending scans, or statements with CONNECT BY clauses.

Reverse key and bitmap indexes do not support the skip scan algorithm.

Oracle9i: New Features for Administrators 10-7

Skip Scanning Example

LANGUAGE and TERRITORY combinations:

LANGUAGE	TERRITORY
ENGLISH	AMERICA
ENGLISH	CANADA
ENGLISH	UK
FRENCH	CANADA
FRENCH	FRANCE
FRENCH	SWITZERLAND
GERMAN	GERMANY
GERMAN	SWITZERLAND
PORTUGUESE	BRAZIL
PORTUGUESE	PORTUGAL

ORACLE

10-8

Copyright © Oracle Corporation, 2001. All rights reserved.

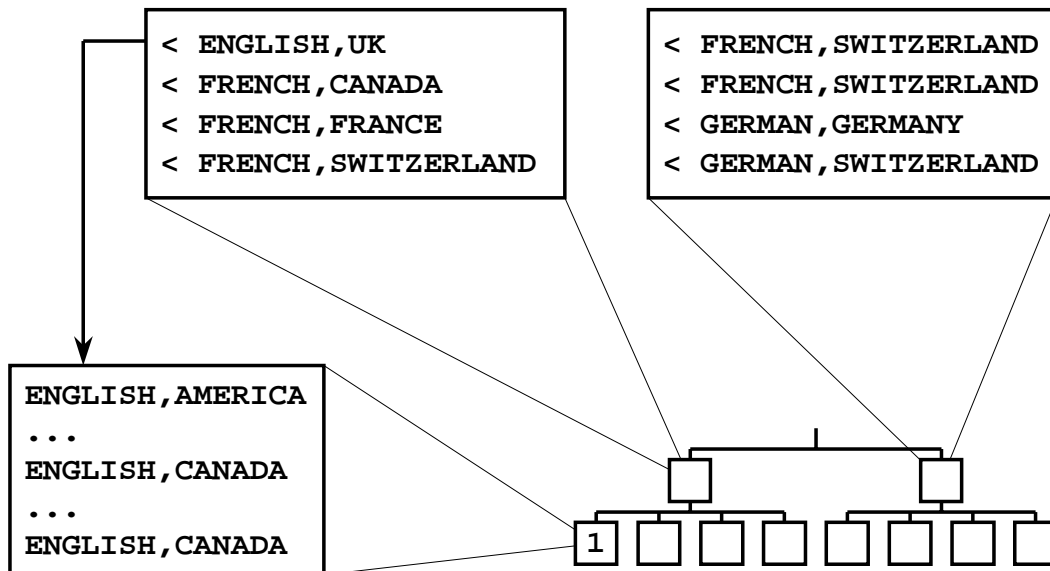
Skip Scanning Example

In the example, suppose a composite index exists on the two columns `LANGUAGE` and `TERRITORY`, with `LANGUAGE` as the prefix column. The data values stored in the underlying table result in the combinations of values shown in the table. Each combination may occur multiple times in the table and the resulting index.

In previous releases without the skip scan algorithm, a query on a value in the `TERRITORY` column would be forced to execute a full table scan or a fast full index scan. If such a query were infrequent, this might be acceptable. If the query were more common, then you might have to add a new index on the `TERRITORY` column alone. This new index, however, could negatively impact the performance of DML on the table.

The skip scan solution provides an improvement without the need for the second index. While not as fast as a direct index look up, the skip scan algorithm is faster than a full table scan in cases where the number of distinct values in the prefix column is relatively low.

Skip Scanning: Search for Switzerland



ORACLE

10-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Skip Scanning Example (continued)

Here is part of the index built on the LANGUAGE and TERRITORY columns as shown on the previous slide. This portion of the index contains two branch blocks each of which points to four leaf blocks. The highest value on the leaf block determines the boundary condition for the branch pointers. For example, the first leaf block referenced by the first branch block contains a number of entries but with only two distinct values, “ENGLISH,AMERICA” and “ENGLISH,CANADA,” both less than “ENGLISH,UK.” The pointer from the branch block entry to this leaf block is shown on the slide.

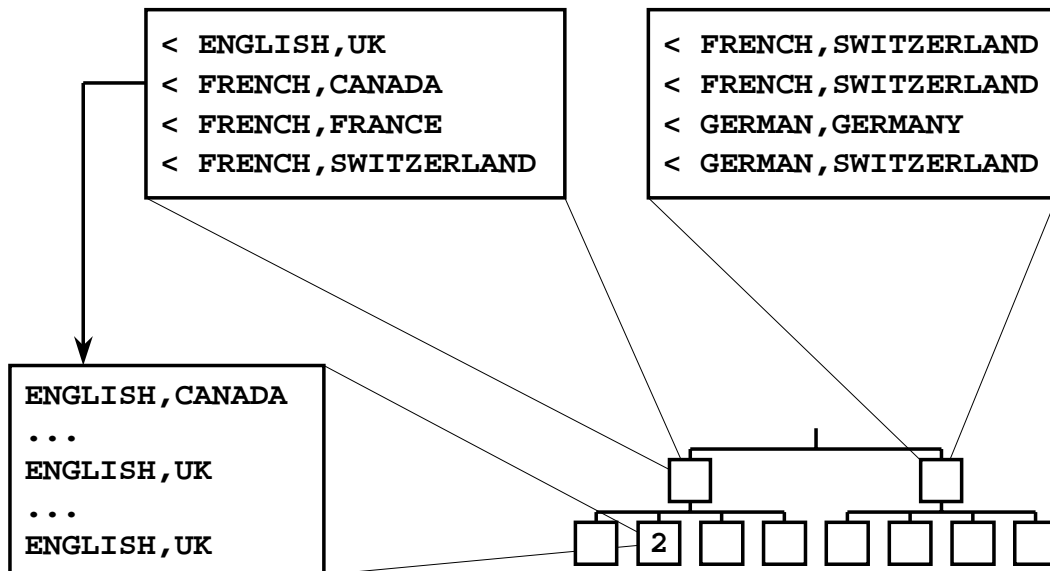
In the next few slides, you will see how the skip scanning algorithm bypasses unneeded leaf blocks when searching for the value “SWITZERLAND” in the TERRITORY column.

First Leaf Block

The index scan begins with the first branch block that indicates that the first leaf block has values less than “ENGLISH,UK.”

The specific values on this first leaf block are unknown at this time, so all of the entries are scanned. At the end of the scan of the first leaf block, the potential value, “ENGLISH,SWITZERLAND,” has not been reached.

Skip Scanning: Search for Switzerland



ORACLE

10-10

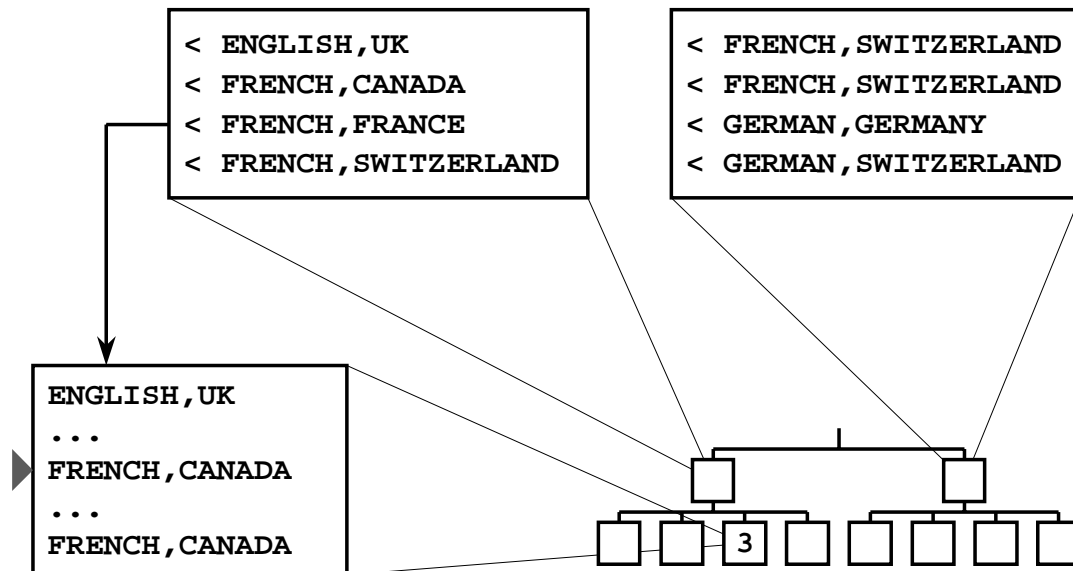
Copyright © Oracle Corporation, 2001. All rights reserved.

Second Leaf Block

The second entry in the first branch block, combined with the last value read on the first leaf block, imply that the entries on the second leaf block are bounded by the values ENGLISH,CANADA and FRENCH,CANADA.

Since there could be a number of LANGUAGE values, including ENGLISH and FINNISH, for example, in this range, the second leaf block is also scanned for any entries with an TERRITORY value of SWITZERLAND.

Skip Scanning: Search for Switzerland



ORACLE

10-11

Copyright © Oracle Corporation, 2001. All rights reserved.

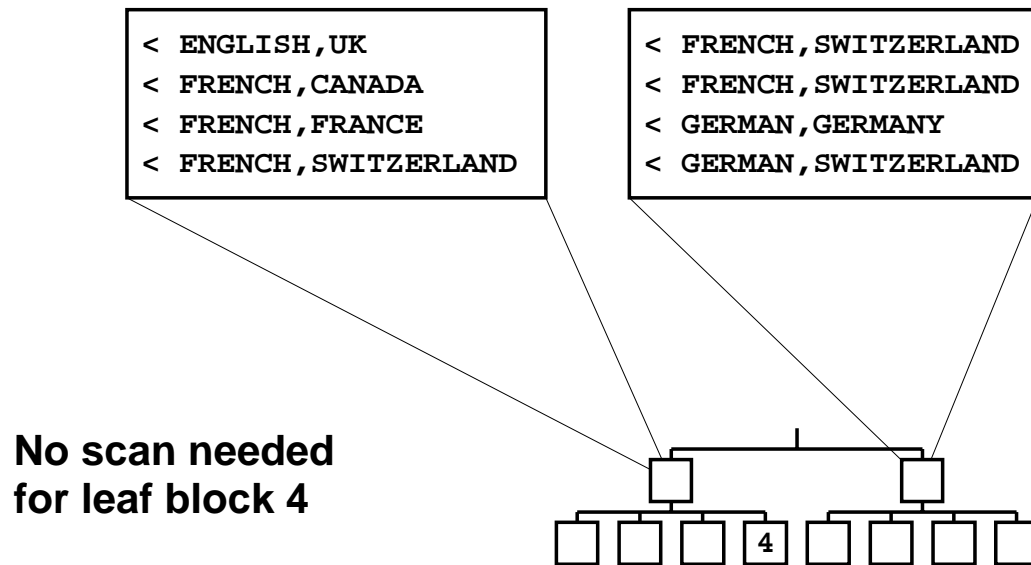
Third Leaf Block

The third entry in the first branch block, combined with the last value read on the second leaf block, imply that entries on the third leaf block are bounded by the values `ENGLISH, UK` and `FRENCH, FRANCE`.

Although the value `ENGLISH, SWITZERLAND` would not be in this range, there is still the possibility of values such as `FINNISH, SWITZERLAND`, for example, appearing on this leaf block.

However, as the third leaf block is scanned, the value `FRENCH, CANADA` is encountered. Because the block does not contain values greater than `FRENCH, FRANCE`, the value `FRENCH, SWITZERLAND` cannot be in this leaf block. Further, there can be no other `LANGUAGE` values besides `FRENCH` and so the remainder of the third leaf block can be skipped.

Skip Scanning: Search for Switzerland



ORACLE

10-12

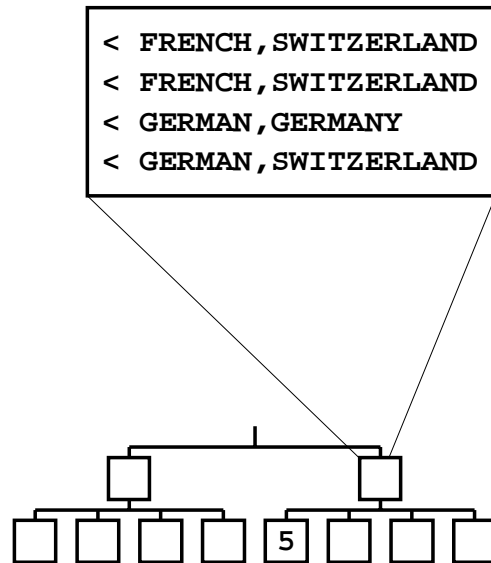
Copyright © Oracle Corporation, 2001. All rights reserved.

Fourth Leaf Block

The fourth entry in the first branch block indicates that the final value on the fourth leaf block must be less than FRENCH,SWITZERLAND. The last value on the third leaf block also has a LANGUAGE value of FRENCH, a fact that can be determined from the previous two branch entries. This implies that the entire block must consist of entries with an LANGUAGE of FRENCH, none of which has an TERRITORY value of SWITZERLAND. The scan therefore skips this entire block.

Skip Scanning: Search for Switzerland

**No scan needed
for leaf block 5**



ORACLE

10-13

Copyright © Oracle Corporation, 2001. All rights reserved.

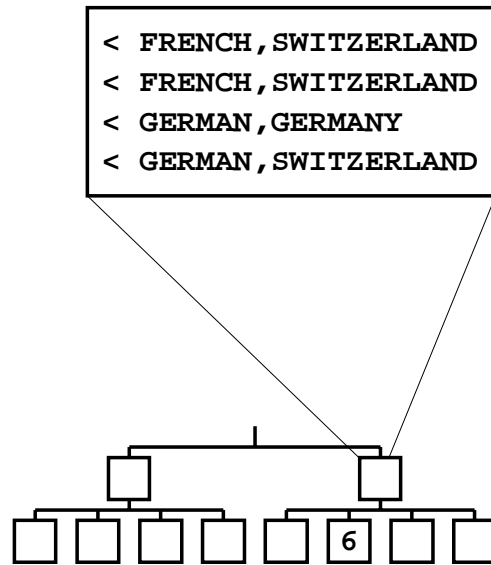
Fifth Leaf Block

The entries on the first branch block have all been read, so the second branch block is read to find the upper possible value on the fifth leaf block. This entry, just like the last entry on the first branch block, implies that the entire fifth leaf block must consist of entries with a LANGUAGE of FRENCH, none of which has a TERRITORY of SWITZERLAND.

The scan therefore skips the fifth leaf block.

Skip Scanning: Search for Switzerland

**No scan needed
for leaf block 6**



ORACLE

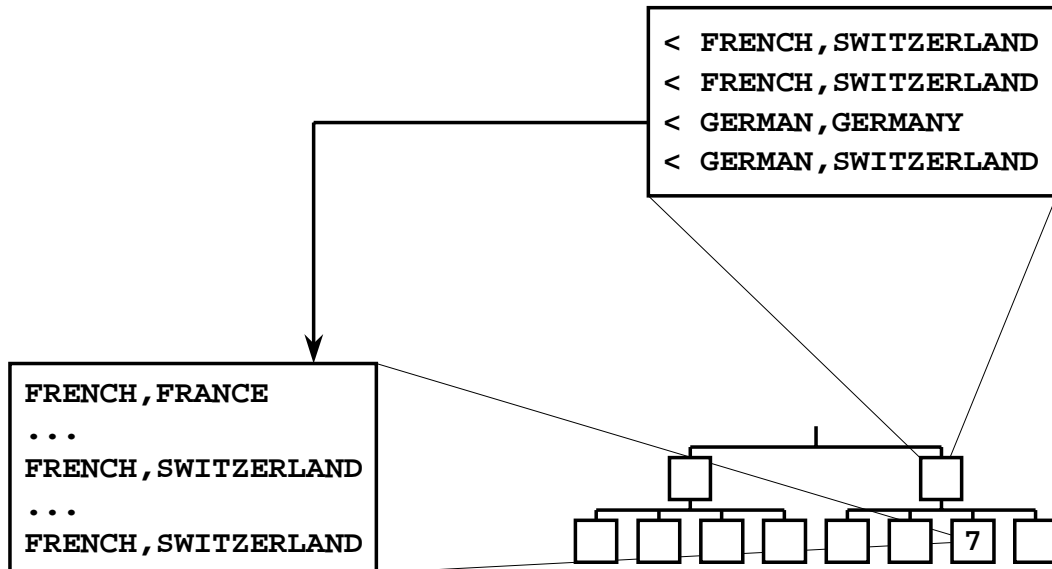
10-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Sixth Leaf Block

The second branch value on the second branch block is exactly the same as the first branch entry. By the same reasoning used for the fifth leaf block, the sixth leaf block is also skipped.

Skip Scanning: Search for Switzerland



ORACLE

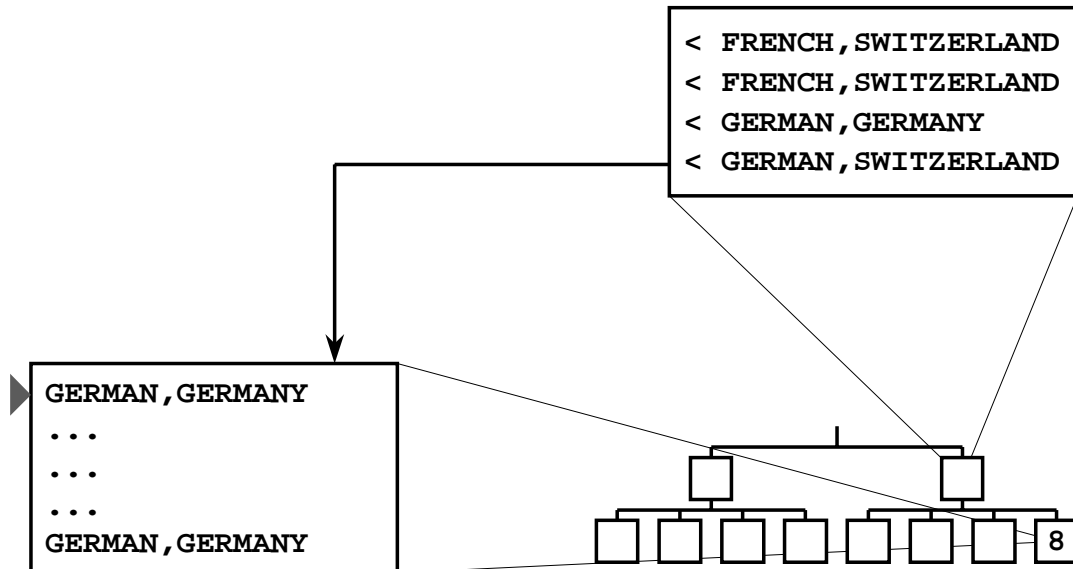
10-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Seventh Leaf Block

The third value on the second branch block indicates that the seventh branch block could contain at least two different LANGUAGE values, bracketed by FRENCH and GERMANY. Any one of these LANGUAGE values could have the required value of SWITZERLAND in the TERRITORY column, except GERMANY, which is excluded by the upper boundary value. The seventh leaf block must therefore be read.

Skip Scanning: Search for Switzerland



ORACLE

10-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Eighth Leaf Block

The final value on the second leaf block indicates that entries on eighth leaf block with a LANGUAGE value of GERMANY will not contain the required value of SWITZERLAND in the TERRITORY column. However, it is not possible to tell if there are more entries with another language value in this eighth leaf block, so the scan reads the block. But, after finding that the first entry has the value of GERMANY in the LANGUAGE column, the scan can skip the rest of the eighth leaf block.

Example Summary

Even in this very small example, you can see that almost half the leaf blocks would be skipped by the index skip scanning algorithm, and the entire index has not been examined. Obviously, different results will be obtained depending on the range and distribution of the values in the index columns. The optimizer uses statistics to determine whether a skip scan retrieval would be more efficient than a full table scan, or other possible retrieval paths, when parsing SQL statements.

Cursor Sharing Enhancements

- Oracle8i introduced the possibility of sharing SQL statements that differ only in literal values.
- Oracle9i extends this feature by limiting it to similar statements only, instead of forcing it.
- Similar: Regardless of the literal value, same execution plan:

```
SQL> SELECT * FROM employees  
2 WHERE employee_id = 153;
```

- Not similar: Possible different execution plans for different literal values:

```
SQL> SELECT * FROM employees  
2 WHERE department_id = 50;
```

ORACLE

Cursor Sharing Enhancements

Oracle8i introduced the possibility to share SQL statements that differ only in literal values. Rather than developing an execution plan each time the same statement—with a different literal value—is executed, the optimizer generate a common execution plan used for all subsequent executions of the statement.

Because only one execution plan is used instead of potential different ones, this feature should be tested against your applications before deciding to enable it or not. That is why Oracle9i extends this feature by sharing only statements considered as similar. That is, only when the optimizer has the guarantee that the execution plan is independent of the literal value used.

Consider a query like this one, where EMPLOYEE_ID is the primary key

```
SQL> SELECT * FROM employees WHERE employee_id = 153;
```

The substitution of any value would produce exactly the same execution plan. It would, therefore, be safe for the optimizer to generate only one plan for different occurrences of the same statement executed with different literal values.

On the other end, assume the same EMPLOYEES table has a wide range of values in its DEPARTMENT_ID column. For example, department 50 could contain over one third of all employees and department 70 could contain just one or two. Given the two queries:

```
SQL> SELECT * FROM employees WHERE department_id = 50;
```

```
SQL> SELECT * FROM employees WHERE department_id = 70;
```

Cursor Sharing Enhancements (continued)

Using only one execution plan for sharing the same cursor would not be safe if you have histogram statistics (and there is skew in the data) on the `department_id` column. In this case, depending on which statement was executed first, the execution plan could contain a full table (or fast full index) scan, or it could use a simple index range scan.

CURSOR_SHARING Parameter

- **CURSOR_SHARING parameter values:**
 - **FORCE**
 - **EXACT (default)**
 - **SIMILAR (new in Oracle9i)**
- **CURSOR_SHARING can be changed using:**
 - **ALTER SYSTEM**
 - **ALTER SESSION**
 - **Initialization parameter files**

ORACLE

10-19

Copyright © Oracle Corporation, 2001. All rights reserved.

CURSOR_SHARING Parameter

The value of the initialization parameter `CURSOR_SHARING` determines how the optimizer will process statements with bind variables:

- **EXACT:** Literal replacement disabled completely
- **FORCE:** Causes sharing for all literals
- **SIMILAR:** Causes sharing for safe literals only

In previous releases, you could choose only the `EXACT` or the `FORCE` option. The `SIMILAR` option is new in Oracle9i. It causes the optimizer to examine the statement to ensure that replacement occurs only for safe literals. In doing this, it can use information about the nature of any available index (unique or non-unique) and statistics collected on the index or underlying table, including histograms.

The value of `CURSOR_SHARING` in the initialization file can be overridden with an `ALTER SYSTEM SET CURSOR_SHARING` or an `ALTER SESSION SET CURSOR_SHARING` command.

Cached Execution Plans

- **The cached execution plans feature enables the Oracle server to preserve the actual execution plan of a cached SQL statement in memory.**
- **After the SQL statement ages out of the Library Cache, the corresponding cached execution plan is removed.**
- **The main benefit of this feature is better diagnosis of query performance.**

ORACLE

New View to Support Cached Execution Plans

- A new dynamic performance view, `V$SQL_PLAN`, stores actual execution plan information for a cached cursor.
- The view contains all of the `PLAN_TABLE` columns, except for the `LEVEL` column, in addition to four new columns.
- Columns that are also present in the `PLAN_TABLE` will have the same value as those in `V$SQL_PLAN`.

ORACLE

10-21

Copyright © Oracle Corporation, 2001. All rights reserved.

New View to Support Cached Execution Plan History

The `V$SQL_PLAN` view stores the actual execution plan information for a cached cursor. The view contains all of the `PLAN_TABLE` columns with the exception of the `LEVEL` column and four new columns. The columns present in the `PLAN_TABLE` have the same values as those in the `V$SQL_PLAN`.

Additional `V$SQL_PLAN` Columns Not Found in the `PLAN_TABLE`:

- `ADDRESS`: Cursor parent handle address
- `HASH_VALUE`: Parent statement hash value in library cache
- `CHILD_NUMBER`: Number using this execution plan
- `DEPTH`: Level of the operation in the tree

New PLAN_HASH_VALUE Column in V\$SQL

Hashed value is obtained from the corresponding execution plan. You can use it for comparisons:

- **Take a snapshot of cursor plan information (`plan_hash_value`) of cached SQL statements**
- **Execute your application again**
- **Find queries whose plans have changed (a different value for `plan_hash_value`)**

ORACLE

10-22

Copyright © Oracle Corporation, 2001. All rights reserved.

New PLAN_HASH_VALUE Column in V\$SQL

A new column `plan_hash_value` is added to the `v$sql` view; it contains a hash value built from the cursor plan information.

This new column should be used to compare cursor plans the same way the `hash_value` column is used to compare cursor SQL texts.

For example, in case you expect changes in execution plans following a particular event (for example, changing the value of a major parameter or migrating to a new release of a database or application) you can:

- Take a snapshot of cursor plan information (including the `plan_hash_value` and `v$sql_plan` content)
- After the event you can find out about queries whose plan has changed; that is, the `plan_hash_value` is different

New First Rows Optimization

```
OPTIMIZER_MODE = {FIRST_ROWS_1  
                  | FIRST_ROWS_10  
                  | FIRST_ROWS_100  
                  | FIRST_ROWS_1000}
```

```
SQL> ALTER SESSION SET  
      2  OPTIMIZER_GOAL = FIRST_ROWS_n
```

```
SQL> SELECT /*+ FIRST_ROWS(x) */ ...  
      2  FROM ...
```

ORACLE

10-23

Copyright © Oracle Corporation, 2001. All rights reserved.

New First Rows Optimization

Oracle9i introduces a new way of doing first rows optimization. The old mode was partially based on heuristics, such as always use an index if possible. These heuristics could sometimes lead to very bad plans.

The new mode is completely cost based and allows optimizing for a particular number of first rows; for example, the first 10 rows.

It is invoked as an argument to the initialization parameter `optimizer_mode` or the session parameter `optimizer_goal`, which allows the following values:

- `first_rows_1`
- `first_rows_10`
- `first_rows_100`
- `first_rows_1000`

In addition, the `FIRST_ROWS` hint now takes a numeric argument that is not limited to the values for the parameter. For instance, you could say `/*+FIRST_ROWS(20)*/`

Without a numeric argument, both the parameter and the hint imply the old first rows behavior, which is retained for backwards compatibility and plan stability.

New Gathering Statistic Estimates

- **DBMS_STATS.AUTO_SAMPLE_SIZE:**
New estimate_percent value
- **New method_opt options:**
 - **REPEAT:** New histogram with same number of buckets
 - **AUTO:** New histogram based on data distribution and application workload
 - **SKEWONLY:** New histogram based on data distribution

```
SQL> EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS( -  
2  ownname           => 'OE', -  
3  estimate_percent => DBMS_STATS.AUTO_SAMPLE_SIZE, -  
4  method_opt        => 'for all columns size AUTO');
```

ORACLE

10-24

Copyright © Oracle Corporation, 2001. All rights reserved.

New Gathering Statistic Estimations

Because the cost-based approach relies on statistics, users should generate statistics for all tables, clusters, and all indexes accessed by SQL statements before using the cost-based approach. If the size and data distribution of the tables change frequently, then users should regenerate these statistics regularly to make sure that the statistics accurately represent the data in the tables.

Exact statistics computation requires enough space to perform scans and sorts of involved objects. If there is not enough space in memory, then temporary space might be required. Thus, it is also possible to compute only estimations in order to reduce resources needed to gather statistics. The difficulty in computing estimated statistics is to find the best sample size. Some statistics are always computed exactly, such as the number of data blocks currently containing data in a table or the depth of an index from its root block to its leaf blocks. Nevertheless, this is not true for all statistics.

With Oracle9i, Oracle Corporation recommends setting the `ESTIMATE_PERCENT` parameter of the `DBMS_STATS` gathering procedures to the new value `DBMS_STATS.AUTO_SAMPLE_SIZE`. This is introduced to maximize performance gains while achieving necessary statistical accuracy avoiding the extremes of collecting inaccurate statistics and wasting valuable time.

New Gathering Statistic Estimations (continued)

Oracle9i introduces some new possible values for the `method_opt` parameters of the `dbms_stats` gathering procedures:

- If the size option is set to `REPEAT` and the column currently has a histogram with `b` buckets, the Oracle server attempts to create a new histogram with `b` buckets. If the column has no histogram, no new statistics are gathered. This option is used to maintain the same “class” of statistics (histogram or no-histogram) when looking at new data.
- If the size is set to `AUTO`, the Oracle server decides to create a histogram based on the data distribution AND the way the column is being used by the application. This means that the Oracle server not only looks at non-uniformity in value repetition counts (skew) but also to non-uniformity in range (sparsity). If the application has yet to be run for a sufficient amount of time to capture the workload involving this column, it would be better to use the `SKEWONLY` option temporarily.
- If the size is set to `SKEWONLY`, the Oracle server decides to create a histogram based solely on the data distribution (regardless of how the application uses the column). This option is useful when gathering statistics for the first time (before the workload has had time to run). Using `SKEWONLY` can add quite a bit of overhead to statistics collection, so Oracle recommends that customers use `AUTO` after the application has run for a while.

The above example shows you how to collect all table, column, and index statistics for the OE schema where the Oracle server decides what the sampling percentage should be and when histograms are necessary (assuming that the workload has run for a while).

Note: The Oracle server captures workload information for a cursor when it is hard parsed. Information is stored in the SGA and regularly flushed to disk. No access to these memory and disk structures is provided in Oracle9i.

New GATHER AUTO Option

- **New value for the OPTIONS parameter of the DBMS_STATS.GATHER_SCHEMA_STATS procedure**
- **Is equivalent of running DBMS_STATS.GATHER_SCHEMA_STATS both with:**
 - GATHER STALE
 - GATHER EMPTY
- **Simplifies statistics gathering at the schema level**

```
SQL> EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS( -  
2  ownname  => 'OE', -  
3  options  => 'GATHER AUTO');
```

ORACLE

10-26

Copyright © Oracle Corporation, 2001. All rights reserved.

New GATHER AUTO Option

A new value for the OPTIONS parameter of the DBMS_STATS.GATHER_SCHEMA_STATISTICS has been added in Oracle9i. The goal of this option is to simplify statistics gathering at the schema level.

Oracle8i introduced the value 'GATHER STALE' in order to collect statistics only on objects that have the MONITORING flag set.

'GATHER AUTO' is equivalent of running DBMS_STATS.GATHER_SCHEMA_STATISTICS procedure both with OPTIONS set to 'GATHER STALE' and 'GATHER EMPTY'.

Optimizer Cost Model Enhancements

- **More meaningful cost estimates are available.**
The **PLAN_TABLE** contains three new columns:
 - **CPU_COST:** Estimated CPU cost of the operation
 - **IO_COST:** Estimated I/O cost of the operation
 - **TEMP_SPACE:** Estimated temporary space (in bytes) used by the operation
- **Includes CPU usage**
- **Accounts for the effect of caching**
- **Accounts for index prefetching**

ORACLE

10-27

Copyright © Oracle Corporation, 2001. All rights reserved.

Optimizer Cost Model Enhancements

The role of a query optimizer is to produce the best performing execution plan for a given query. This process includes selecting access paths for single tables, the join order if more than one table is involved in the query, and the join methods.

Currently, you have a choice between using the rule-based optimizer (RBO) and the cost-based optimizer (CBO). The CBO uses a cost model to choose between alternative access paths, join order, and join methods, while the RBO uses a set of simple rules.

The CBO compares the cost of several alternatives and selects the one with the lowest cost. In addition to the cost model, the CBO uses a size model in order to derive statistics on intermediate tables; for example, cardinality of the result of a join operation.

The cost model uses statistics on the objects manipulated by the query. Those statistics are produced by using the **DBMS_STATS** package, and are stored in the database dictionary.

The quality of the execution plan produced by the optimizer is dependent on the accuracy of the cost model. The Oracle8i version of this models contains several limitations both in terms of accuracy and completeness. For example, the model assumes independence of columns when computing the selectivity of multiple predicates on different columns and it accounts only for I/O activities.

Optimizer Cost Model Enhancements (continued)

The cost model is extended to take into account the following:

- Allow users or developers to convert the cost into more meaningful information. The `PLAN_TABLE` contains three new columns:
 - `CPU_COST`: The CPU cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. The value of this column is proportional to the number of machine cycles required for the operation.
 - `IO_COST`: The I/O cost of the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, this column is null. The value of this column is proportional to the number of data blocks read by the operation.
 - `TEMP_SPACE`: The temporary space, in bytes, used by the operation as estimated by the optimizer's cost-based approach. For statements that use the rule-based approach, or for operations that don't use any temporary space, this column is null.
- Include CPU usage. CPU usage will be estimated for SQL functions and operators.
- Account for the effect of caching on the performance of nested-loops joins
- Account for index prefetching. Index prefetching consists in fetching multiple leaf blocks in a single IO operation

Gathering System Statistics

- **System statistics enable the CBO to use CPU and I/O characteristics.**
- **System statistics must be gathered on a regular basis; this does not invalidate cached plans.**
- **Gathering system statistics equals analyzing system activity for a specified period of time.**
- **New procedures:**
 - `DBMS_STATS.GATHER_SYSTEM_STATS`
 - `DBMS_STATS.SET_SYSTEM_STATS`
 - `DBMS_STATS.GET_SYSTEM_STATS`

ORACLE

10-29

Copyright © Oracle Corporation, 2001. All rights reserved.

Gathering System Statistics

System statistics allow the optimizer to consider a system's I/O and CPU performance and utilization. For each candidate plan, the optimizer computes estimates for I/O and CPU costs. It is important to know the system characteristics to pick the most efficient plan with optimal proportion between I/O and CPU cost.

System CPU and I/O characteristics depend on many factors and do not stay constant all the time. Using system statistics management routines, database administrators can capture statistics in the interval of time when the system has the most common workload. For example, database applications can process OLTP transactions during the day and run OLAP reports at night. Administrators can gather statistics for both states and activate appropriate OLTP or OLAP statistics when needed. This allows the optimizer to generate relevant costs with respect to available system resource plans.

When Oracle generates system statistics, it analyzes system activity in a specified period of time. Unlike table, index, or column statistics, Oracle does not invalidate already parsed SQL statements when system statistics get updated. All new SQL statements are parsed using new statistics. Oracle Corporation highly recommends that you gather system statistics.

The `DBMS_STATS.GATHER_SYSTEM_STATS` routine collects system statistics in a user-defined time frame. You can also set system statistics values explicitly using `DBMS_STATS.SET_SYSTEM_STATS`. Use `DBMS_STATS.GET_SYSTEM_STATS` to verify system statistics.

Gathering System Statistics Example

- **First day:**

```
EXECUTE DBMS_STATS.GATHER_SYSTEM_STATS(  
interval => 120,  
stattab => 'mystats', statid => 'OLTP');
```

- **First night:**

```
EXECUTE DBMS_STATS.GATHER_SYSTEM_STATS(  
interval => 120,  
stattab => 'mystats', statid => 'OLAP');
```

- **Subsequent days:**

```
EXECUTE DBMS_STATS.IMPORT_SYSTEM_STATS(  
stattab => 'mystats', statid => 'OLTP');
```

- **Subsequent nights:**

```
EXECUTE DBMS_STATS.IMPORT_SYSTEM_STATS(  
stattab => 'mystats', statid => 'OLAP');
```

ORACLE

10-30

Copyright © Oracle Corporation, 2001. All rights reserved.

Gathering System Statistics Example

The above example shows database applications processing OLTP transactions during the day and running reports at night.

First, system statistics must be collected during the day. In this example, gathering ends after 120 minutes and is stored in the `mystats` table.

Then, system statistics are collected during the night. Gathering ends after 120 minutes and is stored in the `mystats` table.

Generally, you will use the above syntax to gather the system statistics. In that case, you must be sure, before invoking the `GATHER_SYSTEM_STATS` procedure with the `INTERVAL` parameter specified, to activate job processes using a command like:

```
SQL> alter system set job_queue_processes = 1;
```

Alternatively, you can also invoke the same procedure with different arguments to enable manual gathering instead of using jobs. For syntax information refer to the *Oracle9i Supplied PL/SQL Packages Reference Release 9.0.0*.

If appropriate, you can switch between the statistics gathered. Note that it is possible to automate this process by submitting a job to update the dictionary with appropriate statistics: During the day, a job may import the OLTP statistics for the daytime run, and during the night, another job imports the OLAP statistics for the nighttime run.

Gathering System Statistics Example

- **Start manual system statistics collection in the data dictionary:**

```
SQL> EXECUTE DBMS_STATS.GATHER_SYSTEM_STATS( -  
2 gathering_mode => 'START');
```

- **Generate the workload**
- **End system statistics collection:**

```
SQL> EXECUTE DBMS_STATS.GATHER_SYSTEM_STATS( -  
2 gathering_mode => 'STOP');
```

ORACLE

Gathering System Statistics Example (continued)

The previous example shows how to collect system statistics using jobs by using the internal parameter of the `DBMS_STATS.GATHER_SYSTEM_STATS` procedure. In order to manually collect system statistics, another parameter of this procedure can also be used as shown in the above example.

First, you need to start the system statistics collection, and then, you can end the collecting process at any time after you are sure that a representative workload has been generated on the instance.

The above example collects system statistics directly in the data dictionary.

Summary

In this lesson, you should have learned how to:

- Monitor index usage with the **MONITORING** clause
- Identify a skip scan index access
- Use the **SIMILAR** value of **CURSOR_SHARING**
- Interpret the **V\$SQL_PLAN** view
- Use the **FIRST_ROWS_n** parameter
- Gather system statistics with **DBMS_STATS** package

ORACLE

Practice 10-1 Overview

This practice covers the following topics:

- **Monitor index usage**
- **Use the MONITORING and NOMONITORING clauses of the ALTER INDEX command**
- **Use the V\$OBJECT_USAGE view**

ORACLE

Practice 10-2 Overview

This practice covers the following topics:

- Show how to use the new `SIMILAR` value of the `CURSOR_SHARING` initialization parameter
- Use the new `V$SQL_PLAN` view

ORACLE

11

Scalable Session State Management

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

Objectives

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

- **Explain Oracle Shared Server enhancements**
- **Invoke dedicated external procedure agents**
- **Manage multithreaded heterogeneous service agents**
- **Implement connection pooling with Oracle Call Interface (OCI)**
- **List the benefits of core library improvements**

ORACLE

Oracle Shared Server Improvements

- **Oracle Net Services connection establishment uses a direct handoff to the dispatcher.**
- **It uses a Common event model for both the Oracle9i database and the Net Services network.**
- **New diagnostic features in Performance Manager for Shared Server**

ORACLE

11-3

Copyright © Oracle Corporation, 2001. All rights reserved.

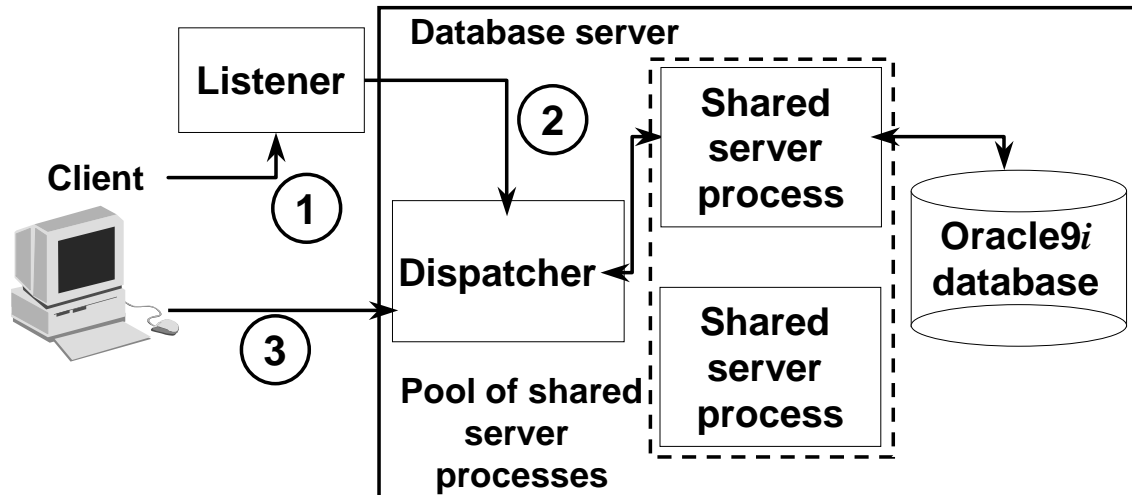
Oracle Shared Server Improvements

Oracle9i Shared Server, previously called multithreaded server (MTS), has the following new features:

- Oracle Net Services connections use a direct handoff to the dispatcher.
- Database and network events require less polling through the use of a common event model.
- New diagnostic details in the Enterprise Manager's Performance Manager for Oracle9i Shared Server

These improvements are discussed in more detail on the following pages.

Connection Establishment: Direct Handoff



ORACLE

11-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Connection Establishment: Direct Handoff

The direct handoff connection method requires less overhead and fewer network messages than in previous releases. Older releases required the listener to obtain the dispatcher's address from the chosen dispatcher and return it to the client who then had to contact the dispatcher to confirm the connection. The client could then continue to communicate with its dispatcher.

In Oracle9i, when a client attempts to connect to the database through a shared server connection, the following events occur, as shown in the diagram:

1. The listener receives the client's request for a shared server connection. This establishes a connection socket between the client and listener.
2. The listener hands connection request, including the connection socket on the listener end, to dispatcher. Now the connection socket is between client and dispatcher.
3. The client communicates directly to the dispatcher using the same connection socket used in step 1. At this point, the client is in direct communication with dispatcher and can send work requests without any further connection processing.

Common Event Model for Database and Network

- **Improvement in the event model**
- **Uses a common event model**
- **Requires no polling between database and network**
- **Reduces CPU consumption and latency**
- **Handles network sessions more efficiently**

ORACLE

11-5

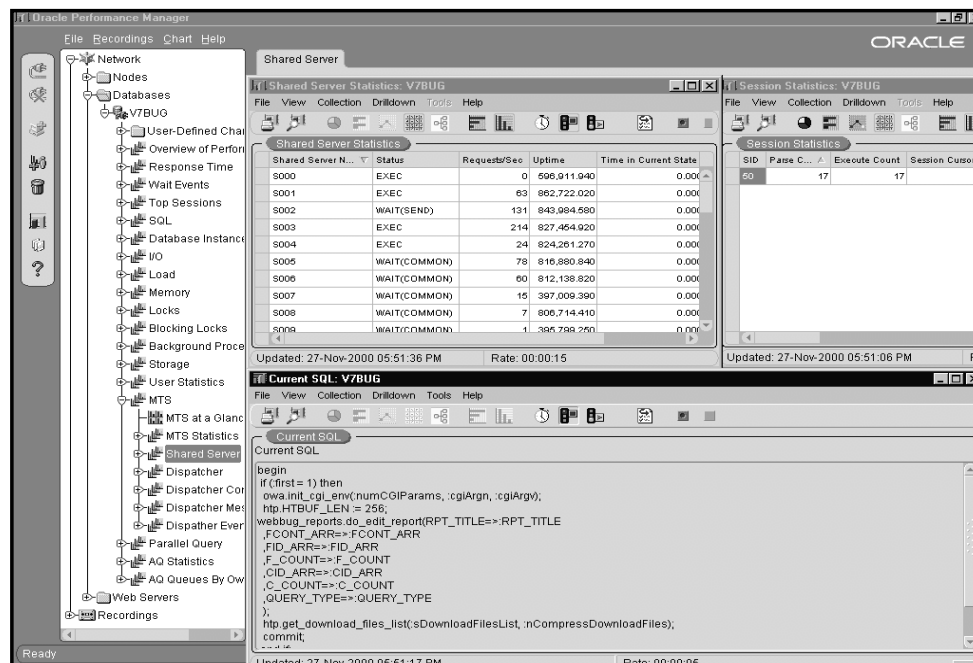
Copyright © Oracle Corporation, 2001. All rights reserved.

Common Event Model for Database and Network

Prior to release Oracle9i, the dispatcher handled network and database events separately and differently. There was no consistency or synchronization among the event notification between these two types of events. The dispatcher would have to check both the network and the database for any newly arriving event. Time and resources were consumed while juggling and balancing these two different event models.

With the new Common Event Model in Oracle9i, the dispatcher can handle event notification more efficiently in the dispatcher and the listener, providing improved performance and scalability with Shared Server.

Performance Manager Monitoring



Performance Manager Real-Time Graphical Monitoring

Performance Manager is part of the Diagnostic Pack under the Oracle9i Enterprise Manager Tool. It provides real-time graphical performance monitoring for Shared Server.

Using Performance Manager, you can monitor the performance behavior of dispatchers, shared servers, and listeners at their individual detailed level, as well as at an overall high level. You can also find out all the session performance information for each shared server and view the SQL statement executed for each session. Performance Manager also provides you with recommendations to fine tune Shared Server related parameters.

External Procedure Agent Enhancements

- External procedures agents allow you to run C programs from a PL/SQL block.
- In Oracle8i, all external procedures use the default agent, `EXTPROC`.
- You have the option of using either `EXTPROC` or a dedicated external procedure, a new type of procedure, in Oracle9i.

ORACLE

11-7

Copyright © Oracle Corporation, 2001. All rights reserved.

External Procedure Agent Enhancements

External procedures allow you to execute C programs using calls from PL/SQL routines. External procedures are commonly used for proprietary code or heavy arithmetic operations which are not handled well by PL/SQL.

In Oracle8i, there is one external procedure process, `EXTPROC`, for each client calling the procedure. Therefore, if 50 clients call an external procedure, there will be 50 copies of the procedure executing. In Oracle9i, the procedure agent allocates only one procedure process for all clients calling that procedure.

Dedicated External Procedure Agents

- **Dedicated external agents**
 - Improve scalability and robustness
 - Allow you to request an agent on any machine
 - Let you specify an alternate or a preferred agent
 - Allow different agents, on different machines, to run the same procedure concurrently
- **External procedures are run through `EXTPROC` by default**
 - Run only on the server
 - Communicate between the agent and the database
 - Use a hard-wired alias so no TNS information has to be passed to the server

ORACLE

11-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Dedicated External Procedure Agents

By grouping external procedures into dedicated agent libraries, load-balancing and scalability are achieved by spreading the load over multiple agents, or even multiple machines.

With Dedicated External procedure agents, the agent to be used to run the external procedures can be specified in PL/SQL at run time. The agent no longer must be on the same machine as the server. It can be on any machine.

Separate agents also increase system robustness. For example, company A is developing a cartridge with external procedures. Also, Company B is developing another cartridge with external procedures. The developers in Company A know that, while they themselves write robust code that never fails, the programmers in Company B are infamous for writing code that almost always aborts and even brings down the operating system. The programmers in Company A, therefore, want their cartridge to run in a separate agent from anyone else (in particular, Company B), so that their code will keep on running even when other developers' codes fail.

Libraries for External Procedures

- **Environment variable specifications in `CREATE LIBRARY` statements allow agents to find external procedures from library instances on different machines.**
- **A string passed to the `CREATE LIBRARY` statement identifies the agent.**
- **`CREATE PROCEDURE` and `CREATE FUNCTION` commands use a string for the agent name to be passed at run time.**
- **Agent names are database link names.**

ORACLE

11-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Libraries for External Procedures

The path name in the `CREATE LIBRARY` statement supports environment variable specifications, to be expanded at run time by the agent. The need for this can be seen by considering the situation in which agents on multiple machines are used to run external procedures from multiple instances of a library. The libraries on the different machines can be in different absolute directories, but as long as they are in the same directory relative to a common root directory (specified by an environment variable), the agents can correctly access the external procedures.

Agent information is passed in a string to the `CREATE LIBRARY` statement to specify an agent. A string is passed to the `CREATE PROCEDURE` and `CREATE FUNCTION` statements specifying the name of an argument in which an agent name will be passed at run time.

The name of the agent is actually the name of a database link. Because it is passed as an argument rather than concatenated to the function name, the problem of local versus remote resolution is avoided.

Example

```
SQL> CREATE DATABASE LINK
  2  agent_link
  3  USING 'agent_tns_alias';
```

```
SQL> CREATE OR REPLACE LIBRARY plib1 IS
  2  '${EP_LIB_HOME}/plib1.so'
  3  AGENT 'agent_link';
  4  /
```

ORACLE

11-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Example

In this example, the agent database link, `agent_link`, will run any dynamically loadable shared libraries (DLLs) in the library `plib1`. If no agent is specified, the default agent `EXTPROC` is used. This ensures backward compatibility with existing external procedure applications. Upon first invocation of any procedure in the library, if the agent is not already running, it is launched. If the specified agent does not exist, it is an error. This is because dedicated agent functionality can be important enough (transaction support, for example) that some sort of default or fallback agent in event of this error (`EXTPROC`, for example) would not typically be acceptable.

The file specification, `${EP_LIB_HOME}/plib1.so`, includes the environment variable `${EP_LIB_HOME}`. Although this format uses UNIX syntax, it executes on any platform. The code expands the variable contained in the curly braces prior to passing it to the operating system. This makes it possible to pass agent arguments to external procedures running on different platforms but with library security that is controlled at the server. Thus, you can code the same `CREATE LIBRARY` command on Windows:

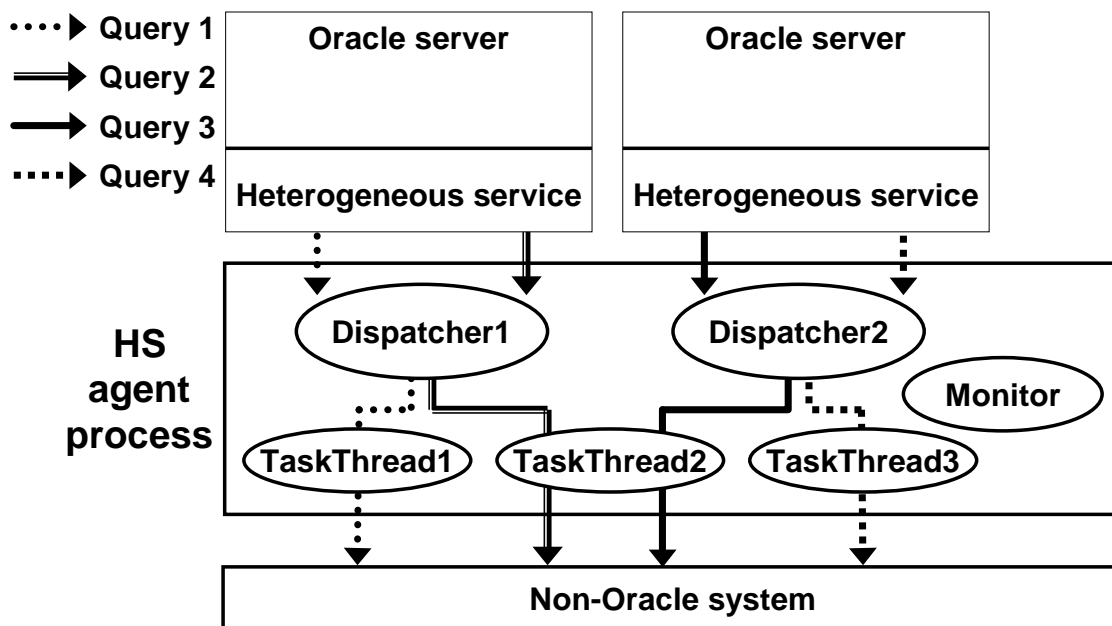
```
SQL> CREATE OR REPLACE LIBRARY plib1 IS
  2  '${EP_LIB_HOME}\plib1.so'
  3  AGENT 'agent_link';
```


Example (continued)

The only difference between the UNIX example, shown on the slide, and this Windows example, is the direction of the slash between the environment variable and the library name.

Note: For more complete examples, see Chapter 10, *Calling External Procedures*, in the manual *Oracle9i Application Developer's Guide – Fundamentals*.

Multithreaded HS Agent Architecture



11-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Multithreaded HS Agent Architecture

The multithreaded agent architecture for HS is similar to the Oracle multithreaded server architecture. The main difference is that it used threads instead of processes.

There are three kinds of threads—a single *monitor* thread, several *dispatcher* threads and several *task* threads. Typically there are many more task threads than dispatcher threads. These three thread types roughly correspond to the Oracle Shared Server's PMON, dispatcher, and shared server processes, respectively. The architecture is shown in the diagram.

The diagram also shows the processing of queries from two different user sessions.

Each session issues two queries. Each request issued by a user-session is shown with a different type of arrow. The queries are passed from HS to the agent where the dispatcher places it on a queue for the task thread to pick up.

All requests from a user-session go through the same dispatcher thread but they can be serviced by different task threads. Several task threads can use the same connection to the non-Oracle system.

The task threads pick up the requests from the queue and perform the necessary operations and place the results on a queue for the dispatcher to pick up.

The dispatcher sends the results of a request back to the server that issued the request.

Multithreaded HS Agent Threads

There are three kinds of threads:

- A single *monitor* thread, responsible for
 - maintaining communication with the listener
 - monitoring the load on the process
 - starting and stopping threads when required
- Several *dispatcher* threads which
 - Communicate with the Oracle server
 - Pass task requests onto the task threads
- Several *task* threads that handle requests from the Oracle processes

ORACLE

11-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Monitor Thread

The monitor thread is responsible for maintaining communication with the listener, monitoring the load on the process, and starting and stopping threads when required. In addition to starting threads and registering dispatchers with the listener, the monitor thread polls the dispatcher threads and sends load information to all the listener processes handling connections to this agent. This way, the listeners can pass incoming connection requests to the least loaded dispatcher. The monitor thread also checks the status of each of the threads it created and, in the event of one dying, cleans up memory and starts a replacement thread.

Dispatcher Thread

The dispatcher threads behave like the dispatcher processes in Oracle's Shared Server architecture. The dispatcher threads handle communication with the database server and pass task requests to the task threads. They accept incoming connections and task requests from Oracle database servers, place incoming requests on a queue for the task threads, and send results to the server that issued the request. Once a session establishes a connection with a dispatcher, all that session's requests use the same dispatcher.

Task Thread

The task threads handle requests from the Oracle processes. The task threads pick up requests from a queue, perform the necessary operations, and place the results on a queue for a dispatcher to pick up.

HS Agent Initialization Parameters

- `max_dispatchers`
- `tcp_dispatchers`
- `max_task_threads`
- `listener_address`
- `shutdown_address`

ORACLE

11-14

Copyright © Oracle Corporation, 2001. All rights reserved.

HS Agent Initialization Parameters

The initialization parameters for HS agents are:

- `max_dispatchers`—maximum number of dispatchers
- `tcp_dispatchers`—number of dispatchers listening on TCP, the rest will use IPC
- `max_task_threads`—number of task threads
- `listener_address`—address on which the listener is listening; needed for registration
- `shutdown_address`—address on which the agent should listen for shutdown messages from `agtctl`

OCI Connection Pooling: Features

- **Allows the creation of a pool of database connections for an application**
- **Provides an interface for the application to specify the optimal, increment, and maximum number of connections in the pool**
- **Provides a mechanism for any OCI call to pickup a server connection from a connection pool**
- **Manages the pool of server connections transparently**

ORACLE

11-15

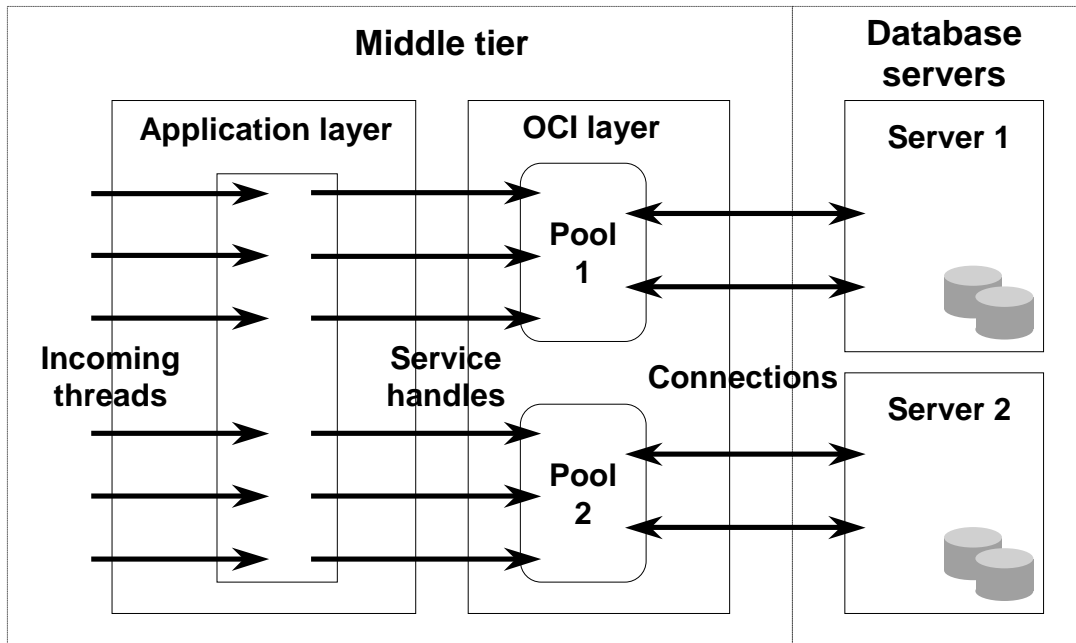
Copyright © Oracle Corporation, 2001. All rights reserved.

OCI Connection Pooling: Features

OCI Connection Pooling allows the creation of a pool of database connections to support an application's requests for data. It is primarily intended for middle tier products developed by Oracle partner companies.

A sample usage of this feature would be in a web application server connected to a back-end Oracle database. A web application server gets several concurrent requests for data from the database server. Typically, it would have to explicitly manage the connections to the database. However, by using this functionality, it can leave that task to OCI. This feature will be useful only for multithreaded applications. The application can create a pool (or set of pools) for each environment handle during initialization. Each thread will have its own service context handle with a user session handle and a virtual server handle.

Usage Model



ORACLE

11-16

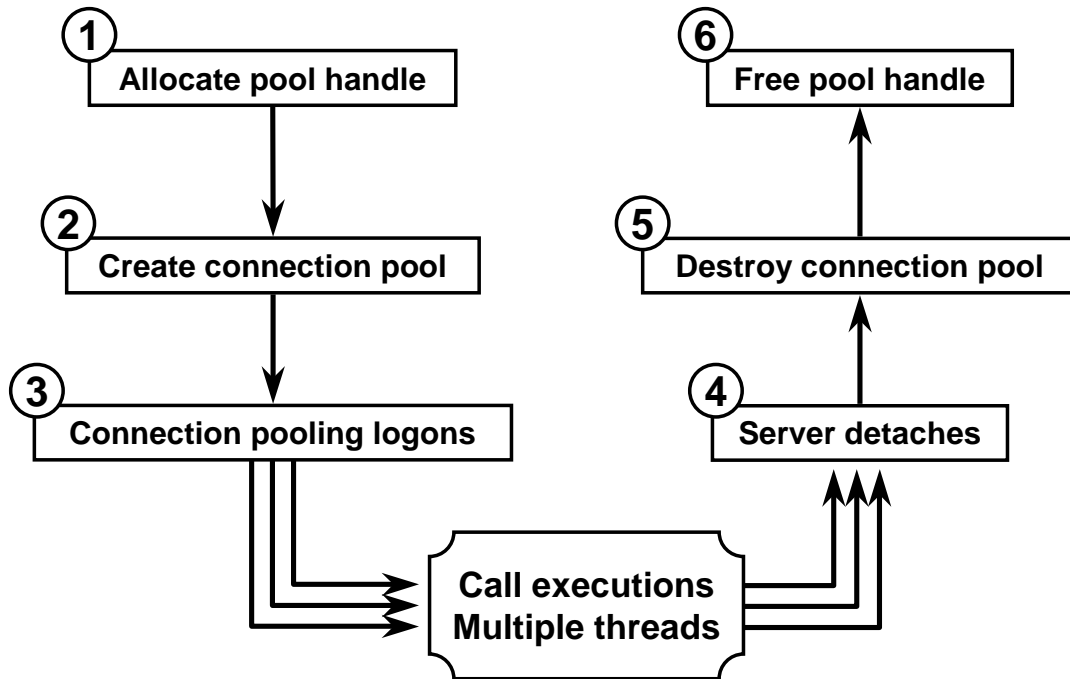
Copyright © Oracle Corporation, 2001. All rights reserved.

Usage Model

The pool creation and the association of the server handles to the pool handle are done in the main thread. The threads then start using the connections in the pool through their respective server handles. So the application need not bind the threads to the server handles (connections), making less use of the connections. The connections are dynamically allotted from the pool, balancing the load properly, thereby increasing the connection utilization.

The OCI connection pools provide more control over the application than network connection pooling. However, OCI connection pooling does not work with transparent application failover (TAF).

Steps Used for OCI Connection Pooling



ORACLE

11-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Steps Used for OCI Connection Pooling

The steps involved in connection pooling for your application are:

1. **Allocate the Pool Handle:** Connection pooling requires that the pool handle `OCI_HTYPE_CPOOL` be allocated by `OCIHandleAlloc()`.
2. **Create the Connection Pool:** The function `OCIConnectionPoolCreate()` initializes the connection pool handle.
3. **Log on to the Database:** Log on to the database for each thread using the call `OCILogon2()`, giving the user name and password, with parameter mode set to `OCI_POOL`. This gives a set of service context handles attached to the pool, which can be used by the threads to process the database calls.
4. **Log off from the Database:** To log off, use `OCILogoff()`.
5. **Destroy Connection Pool:** Use `OCIConnectionPoolDestroy()` to destroy the connection pool.
6. **Free the Pool Handle:** The pool handle is freed using `OCIHandleFree()`.

Core Library Improvements

- **Core library is further modularized.**
- **Reduced dependencies between modules:**
 - **Applications pull in smaller chunks, improving memory usage.**
 - **Smart linkers pull objects into the executable, based upon their atomic level (object files) and their dependencies.**

ORACLE

11-18

Copyright © Oracle Corporation, 2001. All rights reserved.

Core Library Improvements

The libraries that make up the Oracle kernel have been revised to make them more modular. This results in a smaller average size of the kernel library modules in Oracle9i. At the same time, the modules were made as self-contained as possible, reducing the number of dependencies between modules.

The results of these changes improve memory usage because applications acquire smaller modules, and require less memory overall. Also, applications need a smaller number of modules to be linked together to meet dependency requirements. This, too, reduces the overall memory usage by the applications.

Summary

In this lesson, you should have learned how to:

- **Manage Oracle Shared Server**
- **Create libraries, procedures, and C declarations for dedicated external agents**
- **Manage multithreaded heterogeneous service agents**
- **Use OCI for connection pooling**
- **List the benefits of core library improvements**

ORACLE

12

Real Application Clusters

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

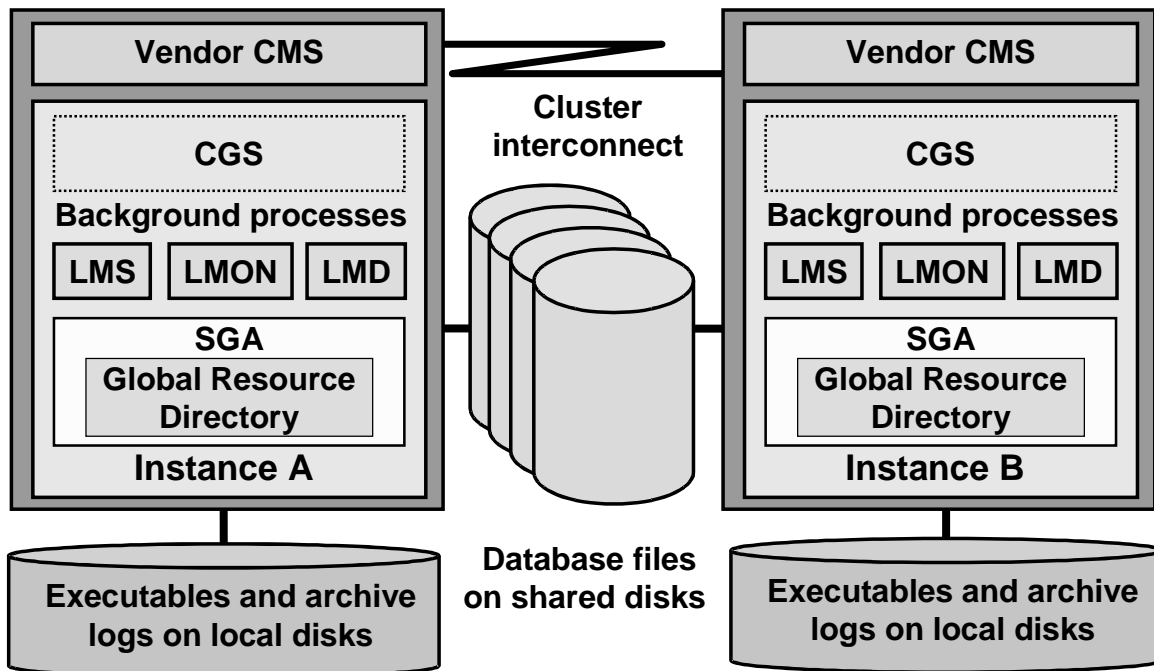
Objectives

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

- **Describe Real Application Clusters architecture**
- **Explain the functions of the Global Cache Service, the Global Enqueue Service, and the Global Resource Directory**
- **Define Cache Fusion block transfer methods**
- **Configure a shared server-side initialization parameter file**
- **Use management tool enhancements**
- **Remove or reset modified initialization parameters**

ORACLE

Real Application Clusters Overview



ORACLE

12-3

Copyright © Oracle Corporation, 2001. All rights reserved.

Real Application Clusters Overview

Oracle9i Real Application Clusters allow multiple instances to execute against the same database. The typical installation involves a cluster of nodes with access to a set of shared disks. A node is defined as the collection of processors, shared memory and disks that runs an instance. A node may have more than one CPU, either in an SMP or a NUMA configuration. The cluster nodes are connected by an interconnect that allows them to share disks and to communicate with each other through cluster management software (CMS).

The shared disks store the database files: data, online redo, and control files. The Oracle9i executables can also be saved on the shared disks although it is advisable to store these on each node's local disks to help ensure availability. Archived redo log files generated by each instance are stored, ideally, on each node's local disks and on a shared device used by each instance.

To ensure the highest level of transfer rates for messages and data, the cluster interconnect should be based on fast interconnect technologies with low latency. This includes high-speed Ethernet and technologies based on the Virtual Interface Architecture (VIA) messaging API and hardware, such as GigaNet and ServerNet.

Vendor-provided CMS monitors the health of processes running in the cluster, and is used by Real Application Clusters for internode messaging and to control the membership of the instances. A layer of Oracle9i software, the Cluster Group Services (CGS), provides an interface to the vendor's CMS and also performs its own instance validation checks.

Note: Other components shown in the graphic are discussed on the next few pages.

Benefits of Real Application Clusters

- **Increased number of users**
- **Scalability**
- **Application speedup**
- **Availability**

ORACLE

12-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits of Real Application Clusters

By using multiple instances, each on its own node, to process against the same database, Real Application Clusters provide the following advantages over single instance databases:

- Increasing user populations can be supported by adding additional nodes when the available nodes reach their session capacity.
- Work can scale (more work can be completed in the same amount of time) when each instance can be optimized to support a maximum workload.
- Some processing, particularly operations that can be executed with parallel components, can complete faster when the work is spread across multiple nodes.
- Applications have higher availability because they can be accessed from any instance. An instance failure on one node does not prevent work from continuing on one or more surviving instances, and transparent application failover enables fast reconnections and resumption of work following an instance failure.

Global Cache Service

Instance A		Instance B		Instance C	
Global Cache Service		Global Cache Service		Global Cache Service	
Re-source	Granted to Instance	Re-source	Granted to Instance	Re-source	Granted to Instance
...
20	A,B,C	21	B	22	A,C
23	A,B	24	A,C	25	C
26	C	27	A,B,C	28	A,B,C
29	B,C	30	B,C	31	B
...

Note: For simplicity, resource status values are not shown

ORACLE

12-5

Copyright © Oracle Corporation, 2001. All rights reserved.

Global Cache Service

A key function of Real Application Clusters is the coordination of resources shared across instances, including block images in their buffer caches. In previous Oracle cluster-enabled software releases, this function was provided through a Distributed Lock Manager which managed block locks, called Parallel Cache Management locks, and other shared resources such as global enqueues. In Oracle9i, database block resources are managed by the Global Cache Service, using the LMS background process, and non-database block resources by the Global Enqueue Service and its LMD process.

The information tracked by the Global Cache and the Global Enqueue Services is stored in the Global Resource Directory using memory from each of the active instances. Each resource is mastered by just one instance at a time, as shown in the graphic for resources 20 through 31. When an instance needs a shared resource, it sends a message to mastering instance across the cluster interconnect using the CGS and the vendor's CMS. The Global Cache and the Global Enqueue Services also send messages to instances, for example, to request them to release resources required by other instances.

For database blocks, the Global Cache Service maintains a list of resource statuses. These resources are acquired and released based on demands of all the instances, independently of transaction and row resources. The resources tracked by the Global Cache Service can be acquired, dropped, or have their statuses converted regardless of the state of any transactions on the related block. Of course, if an instance loses exclusive access to a resource, the block covered by that resource can no longer be changed by that instance until it acquires the resource in exclusive mode again.

Dynamic Resource Remastering

- **Dynamic remastering allows resource masters to be changed without complete reconfiguration.**
- **During processing, lazy dynamic remastering moves resources to the most active instance.**
- **Should an instance leave the group, the background processes only remaster the resources from the departing instance.**
- **Similarly, when a new instance joins the group, the resources are gradually remastered, adapting to cluster workload.**

ORACLE

12-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Dynamic Resource Remastering

In previous releases of Oracle cluster-enabled software, the Distributed Lock Manager lock database was distributed among the available instances. Once assigned to an instance, a lock would not move from its assigned instance until another instance started up or shut down, known as a *node transition*. A node transition required the lock database to be entirely rebuilt across the new set of active instances. While this rebuilding was in progress, access to the Distributed Lock Manager was frozen, disabling new request processing and reducing database availability. Further, when an instance shut down, all information from its portion of the lock database was lost.

In Real Application Clusters, the resources in the Global Resource Directory are remastered (moved to different instances) dynamically. During processing, when particular instances use blocks almost exclusively, the block resources are remastered to those instances. This remastering is done *lazily*, that is, when the resource information can be included with other messages or when the background processes have no pending requests to process, causing no obvious performance overhead.

Dynamic resource remastering also increases database availability during node transitions in Real Application Clusters. Only the resources associated with new or departing instances have to be processed. Both LMS and LMD will continue to process requests for unaffected resources as they perform this work, minimizing the performance impact of node transitions. Further, when an instance shuts down normally (that is, the instance is not aborted), resources are remastered before the shut down completes, preserving the Global Resource Directory contents from that instance.

Global Cache Service Resource Modes

- **Global Cache Service resources have three different modes:**
 - NULL (N)
 - Shared (S)
 - Exclusive (X)
- **The Global Cache Service satisfies requests for resources using a first-in, first-out queue.**
- **The mode of a resource on a block can change independently of transaction status and row level locks associated with the block.**

ORACLE

12-7

Copyright © Oracle Corporation, 2001. All rights reserved.

Global Cache Service Resource Modes

One of the key characteristics of Real Application Clusters databases is their ability to maintain consistent and coherent database buffer caches across instances. This means that if a block is needed by an instance, the Global Cache Service, in conjunction with the LMS process, will ensure that the instance is using the correct version of the block. A block in the buffer cache of an instance must be the current copy, if the instance needs to change its contents, or a valid version to satisfy a read request. A resource can be held by an instance in one of three modes:

- NULL (N)

The NULL mode is the default status for each instance. It indicates that the instance does not currently hold the resource in any mode.

- Shared (S)

The shared resource mode is required for an instance to read the contents of a block, typically to satisfy a query. Multiple instances can hold a shared mode resource on the same block concurrently.

- Exclusive (X)

The exclusive resource mode allows an instance to change the contents of a block covered by the resource. When an instance holds a resource in exclusive mode, all other instances must hold it in NULL mode.

Note: These three resource modes are similar to lock statuses used in Oracle Parallel Server.

Global Cache Service Resource Roles

- **Resources use roles to support Cache Fusion and can be held in one of two roles:**
 - **Local:** Block images associated with the resource are independent of other instances and the Global Cache Service.
 - **Global:** The blocks covered by the resource are dirty in more than one instance and cannot be manipulated independently.
- **A past image is a copy of a dirty block that has been served to another instance.**
- **A past image is maintained until it, or a more recent image of the block, is written to disk.**

ORACLE

12-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Global Cache Service Resource Roles

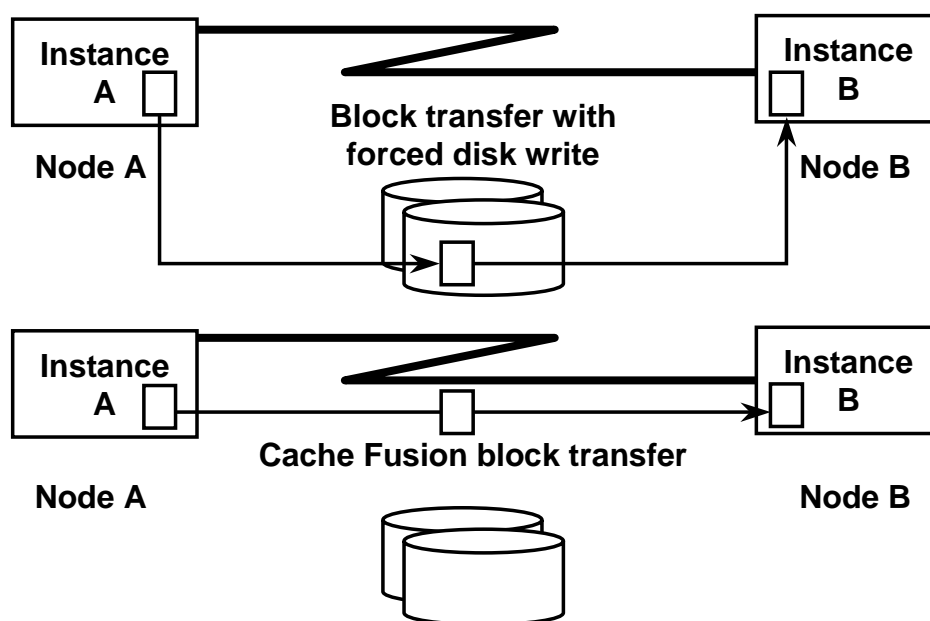
In addition to the resource modes, the Global Cache Service maintains resource *roles* to support Cache Fusion processing. Block resources are held in either a local or a global role.

When a resource is held with a local role, it behaves very similarly to a lock in previous Oracle cluster-enabled software releases. That is, an exclusive mode resource can only be held in one instance at a time and that no other instance can hold that resource in any mode whereas multiple instances can hold the resource in shared mode. Also, the Global Resource Directory does not have to retain any information about a resource being held in NULL mode by an instance.

Global roles allow one instance holds the resource in exclusive mode while other instances concurrently hold that resource in shared or NULL mode. Also, global resource information may be stored in the Global Resource Directory to identify block *past images* even when the resource mode is NULL. With local resources, the Global Cache Service discards the resource allocation information for instances which downgrade a resource to NULL mode.

A block past image is a dirty copy of a block that has been copied to another instance. Past images of a block are retained until the block is finally written to disk. Although the instance with the most current copy of the block typically writes the block to disk, if there have been one or more instance failures, the instance holding the most recent past image performs the block write instead.

Block Transfers



ORACLE

12-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Block Transfers

If a block is only being queried (read) by all the instances, the same image is valid for all the instances and this image is consistent with the on-disk copy. Thus the block image can be read from disk or from another instance.

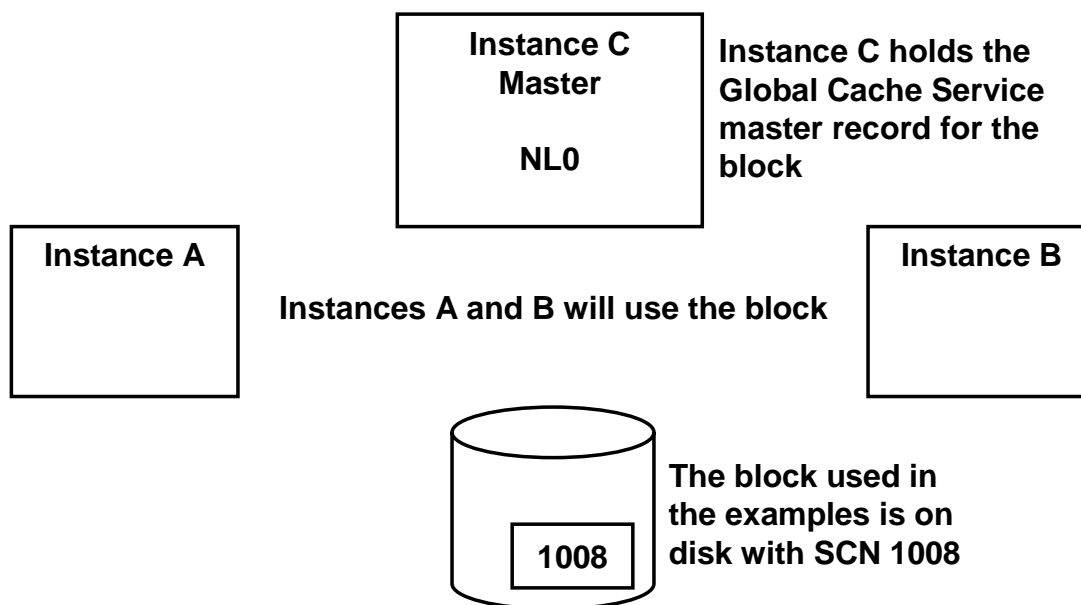
However, when a block has been changed (written) by an instance, another instance must obtain the current block image before it can make its own changes. Similarly, an instance performing a read may need to see changes committed by the writing instance. In these cases, the Global Cache Service instructs the instance holding the current block image to transfer the image to the requesting instance using one of three basic mechanisms:

Block Transfer with Forced Disk Write: This method, often referred to as a block *ping*, was the only block transfer method in Oracle cluster-enabled software releases prior to Oracle8, release 8.1, and was also used in Oracle8i to transfer blocks between writing instances. The block is written to disk by the holding instance and the requesting instance reads the freshly written copy. Although not recommended, you can enable forced disk writes in Oracle9i by setting the `GC_FILES_TO_LOCKS` parameter as discussed later.

Consistent Read Cache Fusion: This algorithm enables the writing instance to prepare a read-consistent image in its own buffer cache and to send this image across the interconnect to the buffer cache of the instance performing the query. A similar algorithm was used in Oracle8i Parallel Server databases.

Cache Fusion Block Transfer: Exclusive to Real Application Clusters, this algorithm passes either clean or dirty block images between instances across the cluster interconnect.

Block Transfer Examples



ORACLE

12-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Block Transfer Examples

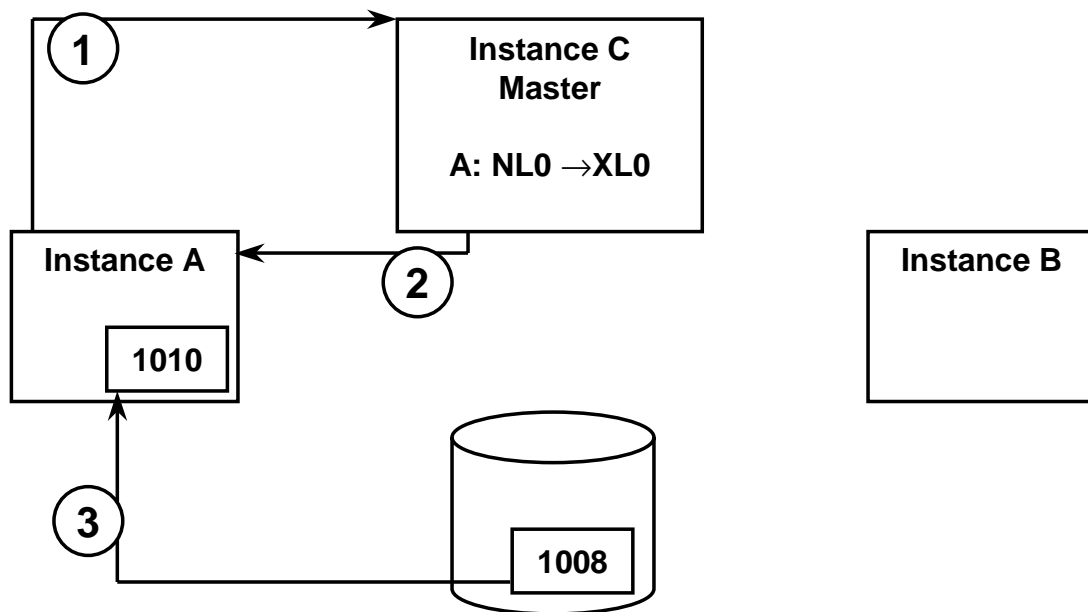
This slide shows the setup used for the following examples which show messages, Global Cache Service resource states, and block movements involved in different types of block transfers. There are three instances, A, B, and C, and a shared drive. For simplicity, the examples use just one block, which is initially shown on the disk with an SCN of 1008. This particular block has its Global Cache Service resource mastered on instance C throughout the examples.

The examples show the resource status as seen by the instances using a three-character notation, for example, SG0, interpreted as follows:

- The first character indicates the mode: N for NULL, S for shared, X for exclusive.
- The second character indicates the role: L for local, G for global
- The third character indicates whether the Global Cache Service knows about a past image: 0 for no, 1 for yes

Resources not held by an instance are in an NL0 (NULL, local, no past image) status. To simplify the examples, an NL0 status is only shown when the resource is involved in a transition to or from another state on a particular instance, for example, NL0→XL0. Also, the graphic does not show the cluster interconnect and disk connections.

Block Transfer Example Setup



ORACLE

12-11

Copyright © Oracle Corporation, 2001. All rights reserved.

Block Transfer Example Setup

To prepare for the upcoming examples, a dirty block image is needed in the buffer cache one of the instances. The following three steps are required for Instance A to acquire the necessary Global Cache Service resource in exclusive mode and a copy of the block:

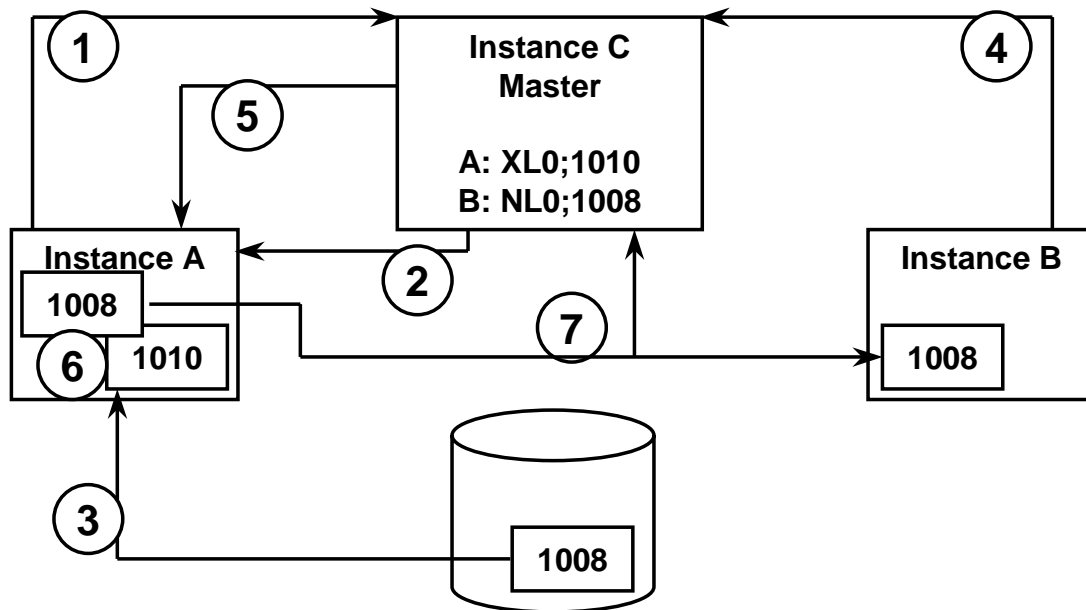
Step 1: Instance A sends a request to the Global Cache Service for an exclusive resource in order to update the block's contents.

Step 2: The Global Cache Service converts the resource status to exclusive for instance A and sends a resource granted message to Instance A.

Step 3: Instance A reads a copy of the block into its buffer cache, protected by the exclusive resource.

Once the block is available, Instance A changes its contents and commits the transaction. In the example, the SCN is updated to 1010 as a result of these actions.

Consistent Read Block Transfer



ORACLE

12-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Example 1: Consistent Read Block Transfer

This example starts following step 3 from the previous slide. That is, the Global Cache Service has already provided an exclusive resource on the block to Instance A. Instance A has read the block from disk and committed changes to the contents, updating the SCN to 1010.

Step 4: Instance B sends a request to the Global Cache Service for a shared resource on the block, including the read-consistent SCN, 1008, which is needed to satisfy the query.

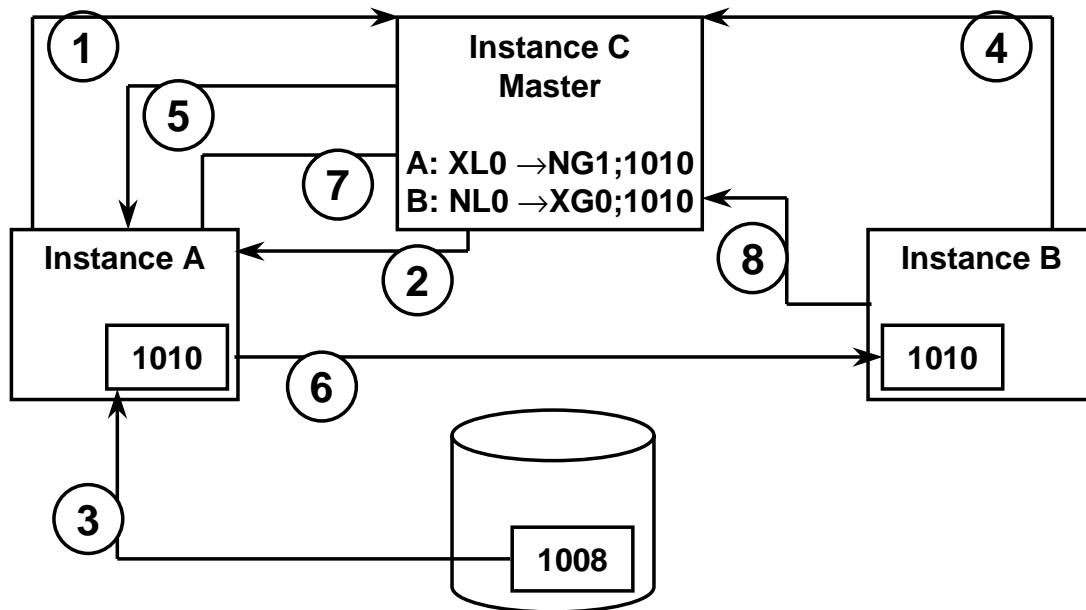
Step 5: The Global Cache Service sends a request to Instance A, the current holder of the exclusive resource, to provide Instance B with a read-consistent copy of the block at SCN 1008.

Step 6: Instance A uses a free buffer in its cache to build a read-consistent copy of the block based on SCN 1008, applying its own undo information to accomplish this.

Step 7: Instance A sends the read-consistent block image, at SCN 1008, to Instance B across the cluster interconnect and notifies the Global Cache Service that it has done this.

With Consistent Read Cache Fusion, the receiving instance does not need a shared or exclusive resource on the block image it obtains from the sending instance. This is because read-consistent blocks are not strictly database blocks; they contain old data that does not need to be written disk. However, the Global Resource Directory maintains a NULL, local resource record for the block image in order to track it.

Cache Fusion Block Transfer



ORACLE

12-13

Copyright © Oracle Corporation, 2001. All rights reserved.

Example 2: Cache Fusion Block Transfer

Again this example starts following step 3 of the example setup: Instance A has obtained an exclusive resource on the block, read the block into memory, and changed the block, updating the SCN to 1010.

Step 4: Instance B sends a request to the Global Cache Service for an exclusive resource in order to update the block's contents.

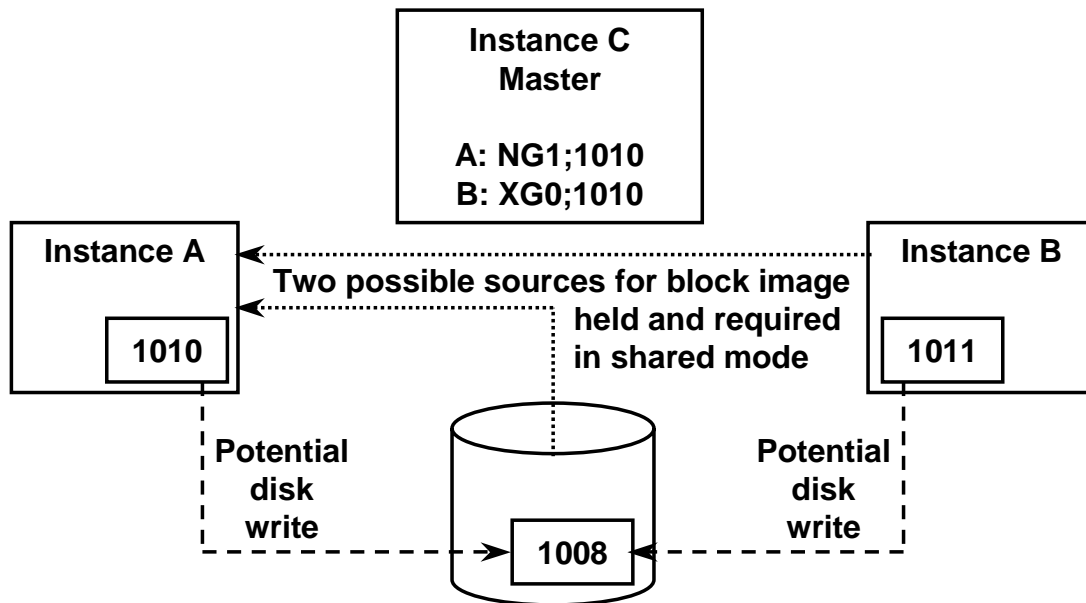
Step 5: The Global Cache Service sends a request to Instance A, the current holder of the exclusive resource, to release its exclusive resource in favor of Instance B.

Step 6: Instance A, realizing that the block is still in its buffer cache, sends an exclusive-keep copy of the block across the interconnect to Instance B.

Step 7: Instance A sends a message to the Global Cache Service to verify that an exclusive-keep copy of the block, with SCN 1010, was sent to Instance B. This causes the Global Cache Service to update the resource status for Instance A to a NULL mode with a global role. The SCN value of 1010 is also stored in the Global Resource Directory along with the past history record flag for Instance A.

Step 8: Instance B sends a message to the Global Cache Service to verify receipt of the block at SCN 1010. The Global Cache Service, in response, adds a resource record for Instance B, indicating that it holds the resource in exclusive mode with a global role (necessary because another instance already has the resource in the global role) with no past history. The Global Resource Directory also stores the SCN (1010) in the resource record for Instance B.

Cache Fusion Block Transfer



ORACLE

12-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Example 2: Cache Fusion Block Transfer (continued)

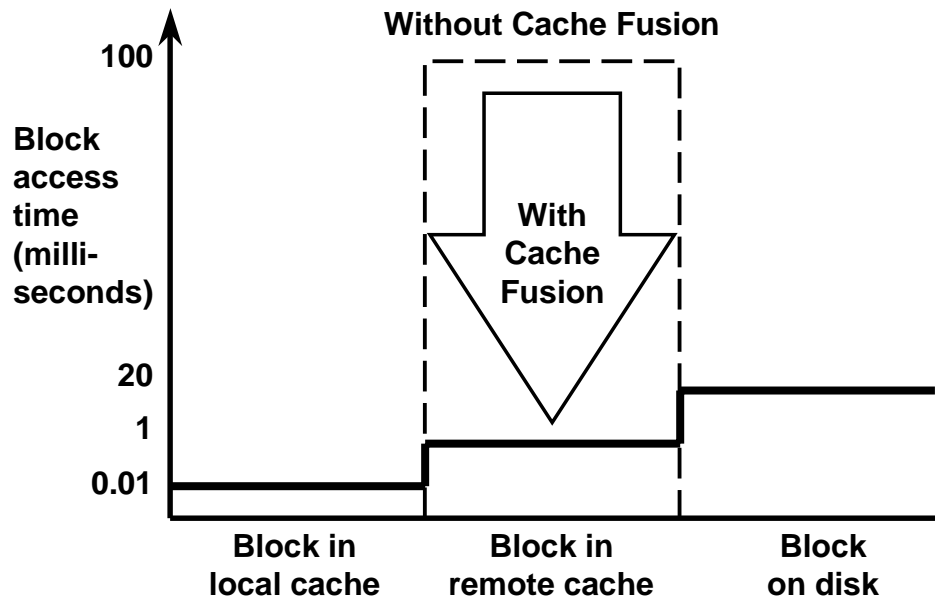
At the end of the process, Instance B can make further changes to the block and, consequently, increment the SCN. In the example, Instance B has increased the SCN to 1011. The past image block is retained at Instance A until the block is finally written to disk. This is done for any number of reasons, including:

- DBWR flushing the buffer cache at either instance to make room for more blocks
- Processing a checkpoint for a redo log switch, a tablespace status change, and so on
- A failure of Instance B, causing the recovery process to use the most current block version at Instance A as a starting image for instance recovery

A request to write a block can come from any instance. It is the responsibility of the Global Resource Directory to identify which instance performs the write to disk when the block is covered by resources in the global role. This will be the instance with the current exclusive resource, if there is one, or else one of the instances with the highest SCN value for the block. Once the write completes, the Global Resource Directory informs all instances with global resources. Each instance frees the related buffer, allowing the Global Resource Directory to remove the past history record for the instance.

A request from an instance to read a block that is held with a shared resource by one or more other instances can also be satisfied with a Cache Fusion block transfer. This is typically faster than a read from disk and does not require any extra messaging to and from the Global Resource Directory once the resource is granted. The resources retain their local role because there are no past images of dirty blocks to track.

Benefits of Cache Fusion



ORACLE

12-15

Copyright © Oracle Corporation, 2001. All rights reserved.

Benefits of Cache Fusion

By allowing any block being held by an instance to be copied to another instance across the cluster interconnect, Cache Fusion allows data to be shared between instances without the overhead of forced disk write block transfers. It even allows an instance to obtain a block already cached in another instance faster than it could read that same block from disk.

Studies on typical clusters with high-speed interconnects regularly demonstrate the levels of speedup shown in the graph. This, in turn, allows applications to run on Real Application Clusters that might not have scaled successfully in previous Oracle cluster-enabled software releases.

High Availability Features

- **Dynamic resource remastering by the Global Cache Service**
- **Concurrent Global Cache Service remastering and instance recovery**
- **CGS contains its own instance validity checking**
- **Use of past images and two-pass redo log recovery**
- **Enhancements to Real Application Clusters Guard (the replacement for Oracle Parallel Fail Safe)**

ORACLE

12-16

Copyright © Oracle Corporation, 2001. All rights reserved.

High Availability Features

Real Application Clusters is Oracle Corporation's premier high availability solution. You can use it to configure alternate instances on a database to be used if the active instances fail due to software or hardware errors. High availability features are incorporated into many of components of the architecture.

- Global Cache Service reconfiguration, discussed earlier, minimizes down time because only resources for one instance need to be remastered after a node transition.
- Due to the relatively small number of resources to be remastered following instance failure, instance recovery by a surviving instance on behalf of the failed instance progresses concurrently with Global Cache Service remastering. In Oracle Parallel Server, instance recovery was deferred during lock database reconfiguration.
- The CGS software contains its own instance status detection algorithms. This enables active instances to monitor each other without relying on the vendor's CMS. Therefore problems are detected quickly, allowing Global Resource Directory reconfiguration and instance recovery to proceed with minimal interruption to service.
- The use of Cache Fusion past images to refresh data files prior to recovery and Oracle9i two pass redo log processing enable faster recovery.
- Real Application Clusters Guard, the Oracle9i replacement for Oracle Parallel Fail Safe, works with more vendor clusters than its predecessor and has improved functionality.

Real Application Clusters Guard

- **High availability database solution for mission critical systems**
- **An integrated feature of Oracle9i Real Application Clusters**
- **Incorporates technology from Oracle9i and the vendor's CMS**
 - **Primary/secondary instance configuration**
 - **Self-contained software packs**
 - **Configuration and management tools**

ACTIVE_INSTANCE_COUNT = 1

ORACLE

12-17

Copyright © Oracle Corporation, 2001. All rights reserved.

Real Application Clusters Guard

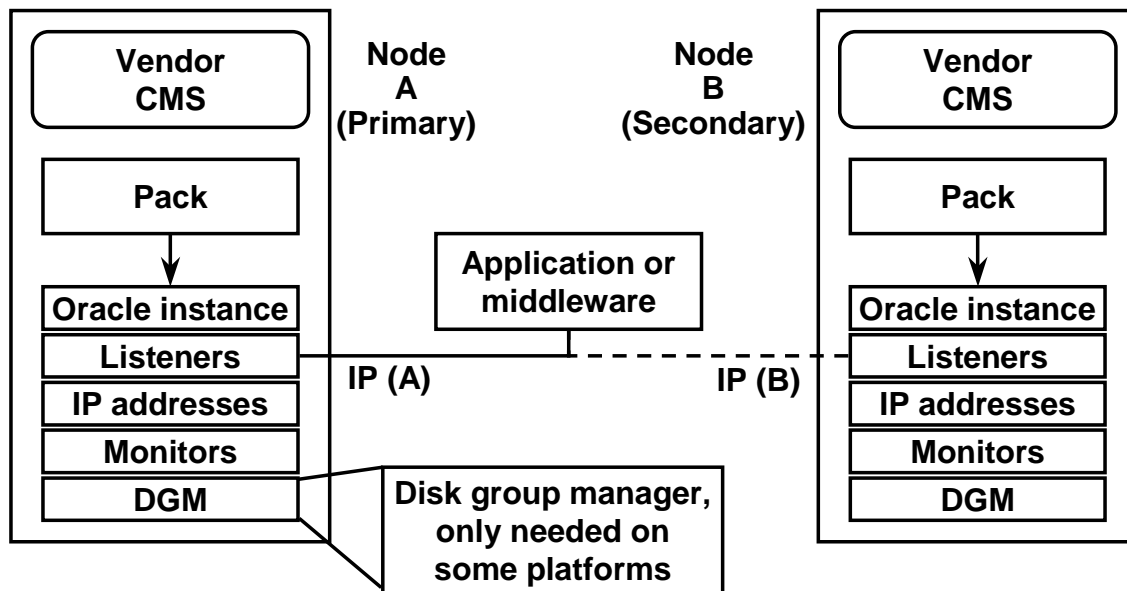
Real Application Clusters Guard provides a high-availability option that minimizes down time in the event of instance or node failure for critical applications. Real Application Clusters Guard combines features of Real Application Clusters and the vendor's CMS. A similar product, Oracle Parallel Fail Safe, was available in earlier releases but only ran on a limited set of clusters (depending on the specific release level) and required a separate installation. In Oracle9i, Real Application Clusters Guard supports a larger variety of clusters and is automatically installed with the Real Application Clusters option.

The typical configuration for Real Application Clusters Guard is a two instance Real Application Clusters database with `ACTIVE_INSTANCE_COUNT = 1` defined in their parameter files. This enables one instance as the primary instance and one as the secondary instance. The primary instance masters the entire Global Resource Directory and is the only instance to allow user connections through Oracle Net Services. The secondary instance takes over the primary role when the primary instance fails.

In addition, the pieces of software necessary to manage an instance are configured into packs. Each pack is a self-contained set of software that can enable and monitor all the components of a Real Application Clusters Guard instance on a node.

The packs and other components of Real Application Clusters Guard are set up and configured with a platform-specific tool. The tool also simplifies deployment of changes in your Real Application Clusters Guard environment.

Real Application Clusters Guard Architecture



ORACLE

12-18

Copyright © Oracle Corporation, 2001. All rights reserved.

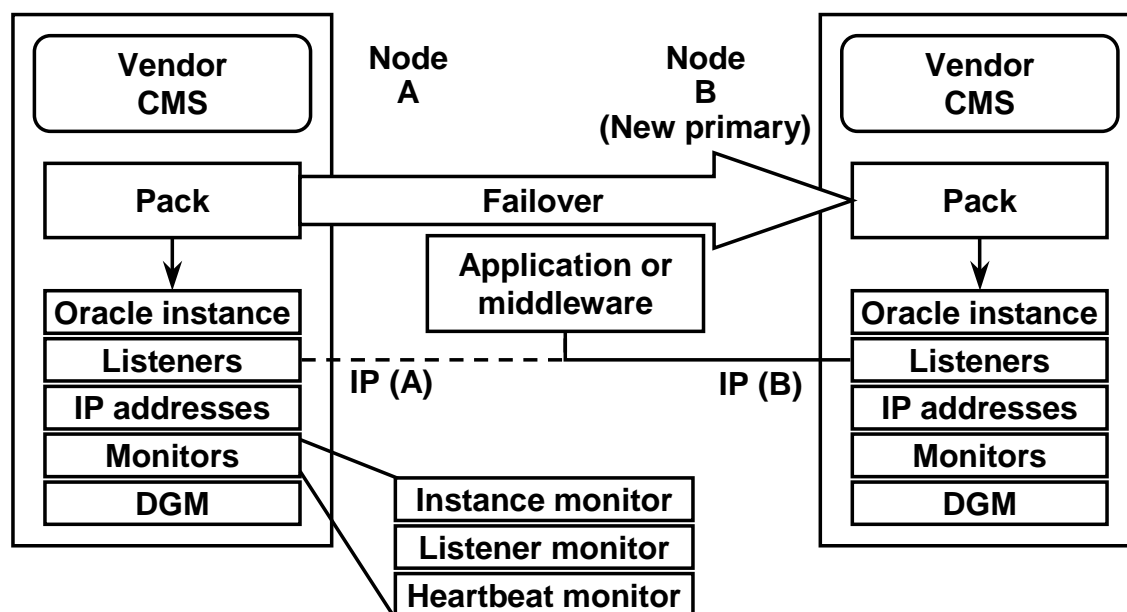
Real Application Clusters Guard Architecture

The graphic shows a two-node Real Application Clusters Guard configuration. Each node executes the vendor's CMS, which in addition to its normal functions (covered earlier in the lesson), is responsible for running and halting scripts automatically upon failover or when you issue the appropriate command.

Each node also contains a pack of software. A pack supports a single instance with access provided through listeners. A pack controls startups, shutdowns, and restarts of the processes under its control. Packs contain the following components:

- A Real Application Clusters instance. In the example shown, Node A is running with the primary instance role and Node B with the secondary instance role.
- One or more listeners to accept Oracle Net connection requests. Public listeners support clients and private listeners support tools such as Oracle Enterprise Manager and Recovery Manager and also provide access for database administration tasks. In the example, the public listener points to the IP address of Node A, the primary node.
- One or more IP addresses. Public IP addresses are relocatable and can be moved between nodes to maintain availability to an active instance. Private IP addresses are static and support connections to private listeners.
- Three monitors, discussed on the next slide.
- A disk group manager (DGM), required only on some platforms, to enable public access to the database disks by the current primary node.

Monitors and Failover



ORACLE

12-19

Copyright © Oracle Corporation, 2001. All rights reserved.

Instance Monitor

The instance monitor detects termination of the local instance and initiates failover to the secondary node or restarts the instance.

Listener Monitor

The listener monitor checks and restarts the listeners on its own node. When the public listener fails to restart, the listener monitor exits, initiating a halt script. Oracle Real Application Clusters Guard either begins failover or restarts the primary instance, depending on the state of the secondary node.

Heartbeat Monitor

The heartbeat monitor checks the availability of the Oracle instance. During normal operation, the heartbeat monitor on each instance updates its own local heartbeat and checks the heartbeat of the other instance. The heartbeat monitor on the primary instance also executes a customer-defined query to test whether the primary instance is capable of work. The local Oracle instance is considered unavailable if the heartbeat monitor fails to complete three consecutive attempts and there are no unusual circumstances, such as instance recovery or unusually large numbers of sessions logging on.

The heartbeat monitor also initiates one kind of failover action: If the primary instance is unavailable and the primary instance role has not resumed normal function on its new node, then the heartbeat monitor initiates takeover. A takeover occurs when the secondary node executes failover of the primary instance role to itself.

Shared Initialization Parameter File

- **Parameters for different instances can be mixed in a single initialization parameter file.**
 - Only one file to maintain and propagate
 - Having all parameter values available in a single location helps reduce errors
- **Use a dot notation with the instance name for instance-specific parameter values.**

ORACLE

12-20

Copyright © Oracle Corporation, 2001. All rights reserved.

Shared Initialization Parameter File

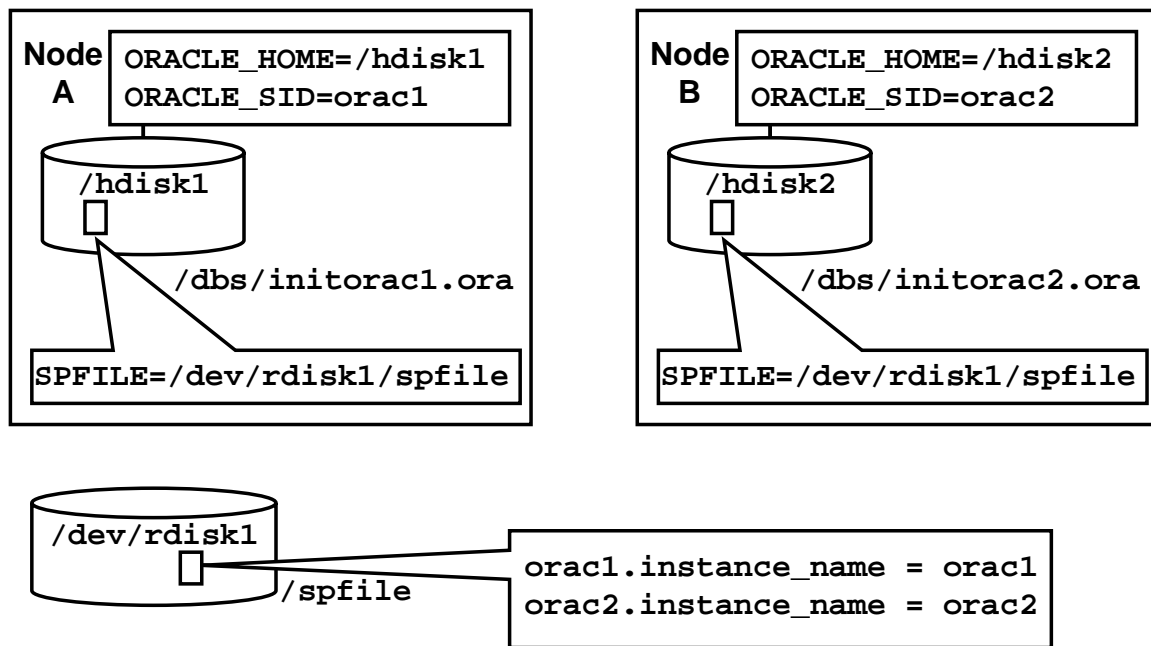
In previous Oracle cluster-enabled software releases, you needed a separate initialization file to assign different parameter values to each instance of your database. At the same time, you had to provide identical values for some parameters across all instances. This required you to maintain multiple parameter files, typically using the `IFILE` parameter to link your common file to the instance-specific files.

In Oracle9i, you can store the parameters for all the instances of a Real Application Clusters database in a single file. This simplifies the management of the instances because you only have to maintain one file. It is also easier to avoid making mistakes, such as changing a value in one file but not another, with all the parameters stored in one place.

To allow different instances to share the same initialization file, parameter entries specific to a particular instance are prefixed with the instance name using a dot notation. For example, to assign different sort area sizes to two instances, `PROD1` and `PROD2`, you could include the following entries in your common parameter file:

```
prod1.sort_area_size = 1048576
prod2.sort_area_size = 524288
```

Shared Server-Side Parameter File



ORACLE

12-21

Copyright © Oracle Corporation, 2001. All rights reserved.

Shared Server-Side Parameter File

When you put the parameters for all your instances in a single initialization file, you need to this file to be available to the process that starts up each instance. If your instances are started automatically as part of the system startup routines, you would need a copy of the file on each node in the cluster. However, you can take advantage of a server side parameter files (SPFILE) in your Real Application Clusters database. To do this, create your shared parameter file, containing parameter values for all of your instances, convert it to an SPFILE, and then store it on shared disk (typically in a raw partition).

In the example, Node A uses /hdisk1 for its ORACLE_HOME directory and supports an instance with a SID value of ORAC1 while Node B uses /hdisk2 for its ORACLE_HOME and runs an instance with a SID of ORAC2. A partition on the raw device, called /dev/rdisk1/spfile, holds the SPFILE.

Each node has an instance-specific initialization file configured containing just one entry

```
spfile = /dev/rdisk1/spfile
```

pointing to the raw partition holding the shared SPFILE. Two entries from the SPFILE are shown in the example, the INSTANCE_NAME parameter for each of the two instances. The values in these parameters follow the recommended naming convention for instances, that is, make them the same as the SID:

```
orac1.instance_name = orac1
```

```
orac2.instance_name = orac2
```

SRVCTL Commands

- **Start all instances with debug output enabled**

```
srvctl start -p dbma -x lsnr -D 3
```

- **Stop the instance but not the listener on node PR1**

```
srvctl stop -p dbma -n pr1 -x lsnr
```

- **Get status of instance DBMA1**

```
srvctl status -p dbma -i dbma1 -s inst
```

- **Delete instance DBMA4 from DBMA**

```
srvctl delete instance -p dbma -i dbma4
```

- **Display all database environment settings**

```
srvctl get env -p dbma
```

ORACLE

12-22

Copyright © Oracle Corporation, 2001. All rights reserved.

SRVCTL Commands

SRVCTL is a tool that lets you manage your Real Application Clusters environment from the command line. It replaces and extends the capabilities of the Parallel Server tool, known as OPSCTL, which supported only the START and STOP subcommands. In comparison to OPSCTL, the START and STOP subcommands have been extended with additional options and new subcommands have been added to support more functionality.

The slide contains examples of SRVCTL commands being used with a database called DBMA and instances called DBMA1, DBMA2, and so on. See the *Oracle9i Real Application Clusters Administration* manual for detailed coverage of the SRVCTL syntax.

Real Application Clusters and OEM

Oracle Enterprise Manager enhancements include:

- **Redo log assignments**
- **Targeted reporting for databases and instances**
- **Extended wizards and tools for cluster databases**
- **Monitoring and event enhancements**
- **Managing Export, Import, load, backup and analyzing tasks**

ORACLE

12-23

Copyright © Oracle Corporation, 2001. All rights reserved.

Real Application Clusters and Oracle Enterprise Manager (OEM)

The following enhancements are included in OEM to support Real Application Clusters:

- **Redo Log Assignment:** Assigns redo log groups to specific threads
- **Report Generation:** Provides targeted reporting for cluster databases and instances
- **Wizards and Tools:** Extends tools (Database wizards, Application Management, Change Management, Database Applications, Diagnostic Pack, Tuning Pack) for cluster databases
- **Events:** Enhances monitoring and events available with OEM and performance packs
- **Jobs:** Provides data management (Export, Import, Load) backup management, and analyze tasks for cluster database and instance targets

OEM Instance Management Provisions

OEM in Oracle9i provides new instance level management:

- **SPFILE handling**
- **Stored configuration**
- **Session handling**
- **Lock details**
- **Resource monitoring**

ORACLE

12-24

Copyright © Oracle Corporation, 2001. All rights reserved.

OEM Instance Management Provisions

These single instance management capabilities were not supported by OEM in clustered databases until Oracle9i:

- **SPFILE Handling:** View and update a server side initialization parameter file (SPFILE)
- **Stored Configuration:** Create, edit, and store multiple startup configurations for each cluster database instance; this eliminates the need to track initialization parameter files for each instance
- **Session Handling:** For each cluster database instance, list the status of connected users, view the latest SQL executed by a session, and kill an unwanted session
- **Lock Details:** For each cluster database instance, list details for currently held user locks (SQL DML enqueues, transaction enqueues, and row level locks) and system locks
- **Resource Monitoring:** For the cluster database, create and modify resource consumer groups and define, modify, and activate resource plans. For each cluster database instance, provide performance statistics of active resource plans

Background Processes

Real Application Clusters background processes replace those used by Oracle Parallel Server

- **The Parallel Server background process BSP is no longer used.**
- **The following background processes have different functions**
 - **LMS: Global Cache Service process**
 - **LMON: Global Enqueue Service Monitor**
 - **LMD: Global Enqueue Service process**

ORACLE

12-25

Copyright © Oracle Corporation, 2001. All rights reserved.

Background Processes

Although some of the background processes used by Real Application Clusters have the same names as those in Oracle Parallel Server, their functions and activities are different from those in previous Oracle cluster-enabled software releases. These differences are discussed below.

- **BSP:** The Block Server Process was introduced in Oracle8i Parallel Server to support Read Consistent Cache Fusion. It does not exist in a Real Application Clusters database where these activities are performed by the LMS process.
- **LMS:** This process was known as the Lock Manager Server Process in Parallel Server. The new LMS process in Real Application Clusters executes as the Global Cache Service Process.
- **LMON:** This process was known as the Lock Manager Monitor in Parallel Server. While Real Application Clusters databases use the same process name, the process is defined as the Global Enqueue Service Monitor.
- **LMD:** There was a Lock Manager process, called LMD0, in Oracle Parallel Server, with a zero at the end of the name implying that multiple copies of the process may be allowed. This process is not used by Real Application Clusters but a new process, called LMD, executes as the Global Enqueue Service.

Initialization Parameters

- **GC_FILES_TO_LOCKS**
 - Uses similar syntax to the identically named parameter in Parallel Server, except that the releasable flag, R, is not needed.
 - Assigning multiblock resources to files by using this parameter turns off Cache Fusion processing.
- **INSTANCE_NUMBER**
 - The default value is 1.
 - Multiple instances must have nondefault values.
- **INSTANCE_NAME**
 - Each instance must have a unique name.
 - Oracle Corporation recommends using the SID defined at the operating system level.

ORACLE

12-26

Copyright © Oracle Corporation, 2001. All rights reserved.

GC_FILES_TO_LOCKS

This parameter assigns Global Cache Service resources to multiple blocks. A similar parameter used in Parallel Server, assigned fixed locks to blocks. An optional flag, the letter R, was used to define these locks as releasable. In Real Application Clusters, all block resources are releasable and the parameter does not support the R syntax, so you may need to change Parallel Server initialization files before using them with Oracle9i.

Setting *file_number=X* or *file_number=X!Y*, where X is nonzero or Y is greater than 1, not only assigns resources to multiple blocks but also disables Cache Fusion processing on the file. To avoid performance problems caused by forced disk writes, only use GC_FILES_TO_LOCKS to assign multiblock resources on files containing read-only or read-mostly data, or data modified primarily by a single instance.

INSTANCE_NUMBER

This parameter provides a unique identification number for each instance to use in its space and instance management routines. The parameter has a default value of 1, requiring you to provide a nondefault value to start a second, or subsequent instance. You may have to change initialization files from your Parallel Server database because they did not need this parameter to be set—unique instance numbers were assigned automatically.

INSTANCE_NAME

This parameter provides a unique name for each instance, if used. You should, ideally, use the same name as the operating system SID value for the instance.

Replaced Initialization Parameters

Parameter in Oracle Parallel Server	Parameter in Real Application Clusters
<code>parallel_server</code>	<code>cluster_database</code>
<code>parallel_server_instances</code>	<code>cluster_database_instances</code>

ORACLE

12-27

Copyright © Oracle Corporation, 2001. All rights reserved.

Replaced Initialization Parameters

Real Application Clusters introduces two other initialization parameters to Oracle cluster software:

- The `CLUSTER_DATABASE` parameter replaces `PARALLEL_SERVER`. This Boolean parameter must be set to True for all instances if you want to start more than one instance in a Real Application Clusters database.
- The `CLUSTER_DATABASE_INSTANCES` parameter replaces `PARALLEL_SERVER_INSTANCES`. This parameter must be set to the highest number of instances you want to start at one time for a Real Application Clusters database.

Before using an initialization parameter from an Oracle Parallel Server database, you will have change these parameters for your Real Application Clusters database.

Obsolete Global Cache Parameters

- **GC_RELEASABLE_LOCKS**
 - All resources are releasable
 - Separate pool of releasable (as distinct from fixed) resources not required
- **GC_ROLLBACK_LOCKS**
 - Not needed to distinguish releasable and fixed resources
 - Segments assigned resources automatically
- **GC_DEFER_TIME**
 - Replaced with a fixed setting
 - No longer adjustable

ORACLE

12-28

Copyright © Oracle Corporation, 2001. All rights reserved.

GC_RELEASABLE_LOCKS

This parameter was used in Oracle Parallel Server to set the number of releasable block locks in addition to the fixed locks defined in GC_FILES_TO_LOCKS. Oracle9i uses only releasable resources for block management so this parameter is invalid and must be removed from any parameter files you want to use with Real Application Clusters databases.

GC_ROLLBACK_LOCKS

Previous Oracle cluster-enabled software releases used this parameter to assign different block locking strategies to individual rollback segments. This initialization parameter is not valid for Real Application Clusters and must be removed from initialization files migrated from Parallel Server databases.

GC_DEFER_TIME

This parameter is invalid in Oracle9i and must be removed from initialization parameter files. You may have used it with your Parallel Server databases to help reduce unnecessary forced disk writes.

Summary

In this lesson, you should have learned how to:

- **Define Global Cache Service resource, Global Enqueue Service, and Global Resource Directory configurations**
- **Enable Cache Fusion**
- **Configure a shared server-side initialization parameter file**
- **Manage databases and instances with enhanced tools**
- **Remove or reset modified initialization parameters**

ORACLE

