
Oracle9i Database Administration Fundamentals I

Volume 1 • Student Guide

D11321GC20
Production 2.0
September 2002
D37261

ORACLE®

Author

Marie St. Gelais

Technical Reviewers

Louise Beijer

Dairy Chan

Trevor Davies

Donna Hamby

Lutz Hartmann

Kuljit Jassar

Patricia Mesa

Sabiha Miri

Howard Ostrow

Caroline Pereda

Andreas Reinhardt

Ajai Sahni

Jaco Verheul

Publisher

Shane Mattimoe

Copyright © Oracle Corporation, 2002. 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.

SQL*Loader, SQL*Net, SQL*Plus, Net8, Oracle Call Interface, Oracle7, Oracle8, Oracle 8i, Developer/2000, Developer/2000 Forms, Designer/2000, Oracle Enterprise Manager, Oracle Parallel Server, PL/SQL, Pro*C, Pro*C/C++, and Trusted Oracle 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

Preface

Introduction

1 Oracle Architectural Components

- Objectives 1-2
- Overview of Primary Components 1-3
- Oracle Server 1-5
- Oracle Instance 1-6
- Establishing a Connection and Creating a Session 1-7
- Oracle Database 1-9
 - Physical Structure 1-10
 - Memory Structure 1-11
 - System Global Area 1-12
 - Shared Pool 1-15
 - Library Cache 1-16
 - Data Dictionary Cache 1-17
 - Database Buffer Cache 1-18
 - Redo Log Buffer 1-21
 - Large Pool 1-22
 - Java Pool 1-24
 - Program Global Area 1-25
 - Process Structure 1-28
 - User Process 1-29
 - Server Process 1-30
 - Background Processes 1-31
 - Database Writer (DBWn) 1-32
 - Log Writer (LGWR) 1-33
 - System Monitor (SMON) 1-34
 - Process Monitor (PMON) 1-35
 - Checkpoint (CKPT) 1-36
 - Archiver (ARCn) 1-37
- Logical Structure 1-39
- Processing SQL Statements 1-42
- Summary 1-44
- Practice 1 Overview 1-45

2 Getting Started with the Oracle Server

- Objectives 2-2
- Database Administration Tools 2-3
- Oracle Universal Installer 2-4
- Starting the Universal Installer 2-5
- Non-Interactive Installation Using Response Files 2-6
- Oracle Database Configuration Assistant 2-9
- Database Administrator Users 2-10
- SQL*Plus 2-12
- Oracle Enterprise Manager 2-13

Oracle Enterprise Manager: Architecture 2-14
Console 2-16
Summary 2-18
Practice 2 Overview 2-19

3 Managing an Oracle Instance

Objectives 3-2
Initialization Parameter Files 3-3
PFILE `initSID.ora` 3-6
Creating a PFILE 3-7
PFILE Example 3-8
SPFILE `spfileSID.ora` 3-9
Creating an SPFILE 3-10
SPFILE Example 3-13
Modifying Parameters in SPFILE 3-14
STARTUP Command Behavior 3-17
Starting Up a Database NOMOUNT 3-19
Starting Up a Database MOUNT 3-20
Starting Up a Database OPEN 3-21
STARTUP Command 3-22
ALTER DATABASE Command 3-25
Opening a Database in Restricted Mode 3-26
Opening a Database in Read-Only Mode 3-29
Shutting Down the Database 3-31
Shutdown Options 3-32
Monitoring an Instance Using Diagnostic Files 3-36
Alert Log File 3-37
Background Trace Files 3-39
User Trace Files 3-40
Enabling or Disabling User Tracing 3-41
Summary 3-43
Practice 3 Overview 3-44

4 Creating a Database

Objectives 4-2
Planning and Organizing a Database 4-3
Optimal Flexible Architecture (OFA) 4-4
Oracle Software and File Locations 4-5
Creation Prerequisites 4-6
Authentication Methods for Database Administrators 4-7
Using Password File Authentication 4-8

- Creating a Database 4-10
- Operating System Environment 4-11
- Database Configuration Assistant 4-12
- Creating a Database Using Database Configuration Assistant 4-13
- Creating a Database Manually 4-16
- Creating a Database Using Oracle Managed Files (OMF) 4-19
- CREATE DATABASE Command 4-23
- Troubleshooting 4-26
- After Database Creation 4-27
- Summary 4-28
- Practice 4 Overview 4-29

5 Using Data Dictionary and Dynamic Performance Views

- Objectives 5-2
- Built-In Database Objects 5-3
- Data Dictionary 5-4
- Base Tables and Data Dictionary Views 5-5
- Creating Data Dictionary Views 5-6
- Data Dictionary Contents 5-7
- How the Data Dictionary Is Used 5-8
- Data Dictionary View Categories 5-9
- Data Dictionary Examples 5-11
- Dynamic Performance Tables 5-12
- Dynamic Performance Examples 5-13
- Administrative Script Naming Conventions 5-15
- Summary 5-17
- Practice 5 Overview 5-18

6 Maintaining the Control File

- Objectives 6-2
- Control File 6-3
- Control File Contents 6-5
- Multiplexing the Control File 6-7
- Multiplexing the Control File When Using SPFILE 6-8
- Multiplexing the Control File When Using PFILE 6-9
- Managing Control Files with OMF 6-10
- Obtaining Control File Information 6-11
- Summary 6-14
- Practice 6 Overview 6-15

7 Maintaining Online Redo Log Files

- Objectives 7-2
- Using Online Redo Log Files 7-3
- Structure of Online Redo Log Files 7-4
- How Online Redo Log Files Work 7-6
- Forcing Log Switches and Checkpoints 7-8
- Adding Online Redo Log File Groups 7-9
- Adding Online Redo Log File Members 7-10
- Dropping Online Redo Log File Groups 7-12
- Dropping Online Redo Log File Members 7-13
- Relocating or Renaming Online Redo Log Files 7-15
- Clearing Online Redo Log Files 7-17
- Online Redo Log File Configuration 7-18
- Managing Online Redo Log Files with OMF 7-20
- Obtaining Group and Member Information 7-21
- Archived Redo Log Files 7-23
- Summary 7-27
- Practice 7 Overview 7-28

8 Managing Tablespaces and Data Files

- Objectives 8-2
- Tablespaces and Data Files 8-3
- Types of Tablespaces 8-4
- Creating Tablespaces 8-5
- Space Management in Tablespaces 8-9
- Locally Managed Tablespaces 8-10
- Dictionary-Managed Tablespaces 8-12
- Migrating a Dictionary-Managed `SYSTEM` Tablespace 8-13
- Undo Tablespace 8-14
- Temporary Tablespaces 8-15
- Default Temporary Tablespace 8-18
- Creating a Default Temporary Tablespace 8-19
- Restrictions on Default Temporary Tablespace 8-22
- Read-Only Tablespaces 8-23
- Taking a Tablespace Offline 8-26
- Changing Storage Settings 8-29
- Resizing a Tablespace 8-31
- Enabling Automatic Extension of Data Files 8-33
- Manually Resizing a Data File 8-36
- Adding Data Files to a Tablespace 8-37

- Methods for Moving Data Files 8-39
- Dropping Tablespaces 8-42
- Managing Tablespaces Using OMF 8-45
- Obtaining Tablespace Information 8-47
- Summary 8-48
- Practice 8 Overview 8-49

9 Storage Structure and Relationships

- Objectives 9-2
- Storage and Relationship Structure 9-3
- Types of Segments 9-4
- Storage Clause Precedence 9-8
- Extent Allocation and Deallocation 9-9
- Used and Free Extents 9-10
- Database Block 9-11
- Multiple Block Size Support 9-12
- Standard Block Size 9-13
- Nonstandard Block Size 9-15
- Creating Nonstandard Block Size Tablespaces 9-17
- Multiple Block Sizing Rules 9-19
- Database Block Contents 9-20
- Block Space Utilization Parameters 9-21
- Data Block Management 9-23
- Automatic Segment-Space Management 9-24
- Configuring Automatic Segment-Space Management 9-26
- Manual Data Block Management 9-27
- Block Space Usage 9-28
- Obtaining Storage Information 9-29
- Summary 9-32
- Practice 9 Overview 9-33

10 Managing Undo Data

- Objectives 10-2
- Managing Undo Data 10-3
- Undo Segment 10-4
- Undo Segments: Purpose 10-5
- Read Consistency 10-6
- Types of Undo Segments 10-7
- Automatic Undo Management: Concepts 10-9

- Automatic Undo Management: Configuration 10-10
- Automatic Undo Management: Initialization Parameters 10-11
- Automatic Undo Management: `UNDO` Tablespace 10-12
- Automatic Undo Management: Altering an `UNDO` Tablespace 10-14
- Automatic Undo Management: Switching `UNDO` Tablespaces 10-16
- Automatic Undo Management: Dropping an `UNDO` Tablespace 10-18
- Automatic Undo Management: Other Parameters 10-21
- Undo Data Statistics 10-23
- Automatic Undo Management: Sizing an `UNDO` Tablespace 10-24
- Automatic Undo Management: Undo Quota 10-26
- Obtaining Undo Segment Information 10-27
- Summary 10-29
- Practice 10 Overview 10-30

11 Managing Tables

- Objectives 11-2
- Storing User Data 11-3
- Oracle Built-in Data Types 11-6
 - `ROWID` Format 11-10
- Structure of a Row 11-12
- Creating a Table 11-13
- Creating a Table: Guidelines 11-17
- Creating Temporary Tables 11-18
- Setting `PCTFREE` and `PCTUSED` 11-19
- Row Migration and Chaining 11-20
- Changing Storage and Block Utilization Parameters 11-21
- Manually Allocating Extents 11-24
- Nonpartitioned Table Reorganization 11-25
- Truncating a Table 11-26
- Dropping a Table 11-27
- Dropping a Column 11-29
- Renaming a Column 11-31
- Using the `UNUSED` Option 11-33
- Obtaining Table Information 11-35
- Summary 11-37
- Practice 11 Overview 11-38

12 Managing Indexes

- Objectives 12-2
- Classification of Indexes 12-3
- B-Tree Index 12-5

- Bitmap Indexes 12-7
- Comparing B-Tree and Bitmap Indexes 12-9
- Creating B-Tree Indexes 12-10
- Creating Indexes: Guidelines 12-13
- Creating Bitmap Indexes 12-15
- Changing Storage Parameters for Indexes 12-18
- Allocating and Deallocating Index Space 12-20
- Rebuilding Indexes 12-21
- Rebuilding Indexes Online 12-23
- Coalescing Indexes 12-24
- Checking Index Validity 12-25
- Dropping Indexes 12-27
- Identifying Unused Indexes 12-29
- Obtaining Index Information 12-30
- Summary 12-31
- Practice 12 Overview 12-32

13 Maintaining Data Integrity

- Objectives 13-2
- Data Integrity 13-3
- Types of Constraints 13-5
- Constraint States 13-6
- Constraint Checking 13-8
- Defining Constraints Immediate or Deferred 13-9
- Primary and Unique Key Enforcement 13-10
- Foreign Key Considerations 13-11
- Defining Constraints While Creating a Table 13-13
- Guidelines for Defining Constraints 13-17
- Enabling Constraints 13-18
- Renaming Constraints 13-23
- Using the `EXCEPTIONS` Table 13-25
- Obtaining Constraint Information 13-28
- Summary 13-31
- Practice 13 Overview 13-32

14 Managing Password Security and Resources

- Objectives 14-2
- Profiles 14-3
- Password Management 14-5
- Enabling Password Management 14-6
- Password Account Locking 14-7
- Password Expiration and Aging 14-8
- Password History 14-9

- Password Verification 14-10
- User-Provided Password Function 14-11
- Password Verification Function `VERIFY_FUNCTION` 14-12
- Creating a Profile: Password Settings 14-13
- Altering a Profile: Password Setting 14-17
- Dropping a Profile: Password Setting 14-19
- Resource Management 14-21
- Enabling Resource Limits 14-22
- Setting Resource Limits at Session Level 14-23
- Setting Resource Limits at Call Level 14-24
- Creating a Profile: Resource Limit 14-25
- Managing Resources Using the Database Resource Manager 14-28
- Resource Plan Directives 14-31
- Obtaining Password and Resource Limit Information 14-33
- Summary 14-35
- Practice 14 Overview 14-36

15 Managing Users

- Objectives 15-2
- Users and Security 15-3
- Database Schema 15-5
- Checklist for Creating Users 15-6
- Creating a New User: Database Authentication 15-7
- Creating a New User: Operating System Authentication 15-10
- Changing User Quota on Tablespaces 15-12
- Dropping a User 15-14
- Obtaining User Information 15-16
- Summary 15-17
- Practice 15 Overview 15-18

16 Managing Privileges

- Objectives 16-2
- Managing Privileges 16-3
- System Privileges 16-4
- System Privileges: Examples 16-5
- Granting System Privileges 16-6
- `SYSDBA` and `SYSOPER` Privileges 16-9
- System Privilege Restrictions 16-10
- Revoking System Privileges 16-11
- Revoking System Privileges with the `ADMIN OPTION` 16-13
- Object Privileges 16-14
- Granting Object Privileges 16-15
- Revoking Object Privileges 16-18
- Revoking Object Privileges with `GRANT OPTION` 16-21

Obtaining Privileges Information 16-22
Summary 16-23
Practice 16 Overview 16-24

17 Managing Roles

Objectives 17-2
Roles 17-3
Benefits of Roles 17-4
Creating Roles 17-5
Predefined Roles 17-7
Modifying Roles 17-8
Assigning Roles 17-10
Establishing Default Roles 17-13
Application Roles 17-15
Enabling and Disabling Roles 17-16
Revoking Roles from Users 17-19
Removing Roles 17-21
Guidelines for Creating Roles 17-23
Guidelines for Using Passwords and Default Roles 17-24
Obtaining Role Information 17-25
Summary 17-26
Practice 17 Overview 17-27

18 Auditing

Objectives 18-2
Auditing 18-3
Auditing Guidelines 18-4
Auditing Categories 18-6
Database Auditing 18-8
Auditing Options 18-10
Auditing User `SYS` 18-12
Obtaining Auditing Information 18-13
Obtaining Audit Records Information 18-14
Summary 18-15
Practice 18 Overview 18-16

19 Loading Data into a Database

Objectives 19-2
Data Loading Methods 19-3
Direct Load 19-4
Serial Direct Load 19-6
Parallel Direct Load 19-7
SQL*Loader 19-9
Using SQL*Loader 19-11

- SQL*Loader Control File 19-13
- Control File Syntax Considerations 19-17
- Input Data and Data Files 19-18
- Logical Records 19-22
- Loading Methods 19-23
- Comparing Direct and Conventional Path Loads 19-26
- Parallel Direct Path Load 19-28
- Data Conversion 19-29
- Discarded or Rejected Records 19-30
- Log File Contents 19-34
- SQL*Loader Guidelines 19-36
- Summary 19-37
- Practice 19 Overview 19-38

20 Using Globalization Support

- Objectives 20-2
- Globalization Support Features 20-3
- Encoding Schemes 20-5
- Database Character Sets and National Character Sets 20-8
- Guidelines for Choosing an Oracle Database Character Set 20-9
- Guidelines for Choosing an Oracle National Character Set 20-11
- Choosing a Unicode Solution:Unicode Database 20-12
- Choosing a Unicode Solution:Unicode Data Type 20-13
- Specifying Language-Dependent Behavior 20-15
- Specifying Language-Dependent Behavior for the Server 20-16
- Dependent Language and Territory Default Values 20-17
- Specifying Language-Dependent Behavior for the Session 20-19
- Linguistic Sorting 20-23
- NLS Sorting 20-24
- Using NLS Parameters in SQL Functions 20-27
- Linguistic Index Support 20-31
- Import and Loading Data Using NLS 20-32
- Using NLS Parameters in SQL Functions 20-33
- Obtaining Character Set Information 20-34
- Obtaining NLS Settings Information 20-35
- Summary 20-39
- Practice 20 Overview 20-40

Appendix A: How to Create an Oracle9i Database in a UNIX Environment

Appendix B: Manually Managing Undo Data (Rollback Segments)

Appendix C: Practice Solutions for SQL*Plus

Appendix D: Practice Solutions for Oracle Enterprise Manager

Preface

Profile

This course is designed to give the Oracle database administrator (DBA) a firm foundation in basic administrative tasks. The primary goal of this course is to give the DBA the necessary knowledge and skills to set up, maintain, and troubleshoot an Oracle database. This course has been designed for junior database administrators, technical support analysts, system administrators, application developers, MIS managers, and other Oracle user.

This preface covers the following sections:

- Before You Begin This Course
- Prerequisites
- How This Course Is Organized
- Related Publications
- Typographic Conventions

Before You Begin This Course

The specific skills you as a participant must have in order to derive the maximum value from attending this course are:

- Familiarity with relational database concepts
- Thorough knowledge of SQL, SQL*Plus and OS commands (UNIX and NT)
- Basic operating system knowledge
- Some working experience with the Oracle environment

Prerequisites

- Introduction to Oracle9i: SQL Basics

How This Course Is Organized

Oracle9i Database Administration Fundamentals I is an instructor-led course featuring lecture and hands-on exercises. This course also allows the capability to accomplish the practices using SQL*Plus or the Oracle Enterprise Manager (OEM). In addition, this course contains clearly defined objectives designed to support preparation for the Oracle Certified Professional examination.

Related Publications

Oracle Publications

Title	Part Number
Oracle9i Backup and Recovery Concepts	A96519-01
Oracle9i Database Administrator's Guide	A96521-01
Oracle9i Database Concepts	A96524-01
Oracle9i Database Error Messages	A96525-01
Oracle9i Database New Features	A96531-01
Oracle9i Database Reference	A96536-01
Oracle9i Database Utilities	A96652-01
Oracle9i Enterprise Manager Administrator's Guide	A96670-01
Oracle9i Enterprise Manager Concepts Guide	A96674-01
Oracle9i Enterprise Manager Configuration Guide	A96673-01
Oracle9i Net Services Administrator's Guide	A96580-01
Oracle9i Net Services Reference Guide	A96581-01
Oracle9i Recovery Manager Reference	A96565-01
Oracle9i Recovery Manager User's Guide	A96566-01
Oracle9i SQL Reference	A96540-01
Oracle9i User-Managed Backup and Recovery Guide	A96572-01

Additional Publications

- System release bulletins
- Installation and user's guides
- *read.me* files
- International Oracle User's Group (IOUG) articles
- *Oracle Magazine*

Typographic Conventions

Typographic Conventions in Text

Convention	Element	Example
Bold italic	Glossary term (if there is a glossary)	The <i>algorithm</i> inserts the new key.
Caps and lowercase	Buttons, check boxes, triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger to the ORD block. Open the Master Schedule window.
Courier new, case sensitive (default is lowercase)	Code output, directory names, filenames, passwords, pathnames, URLs, user input, usernames	Code output: <code>debug.set ('I', 300);</code> Directory: <code>bin (DOS), \$FMHOME (UNIX)</code> Filename: Locate the <code>init.ora</code> file. Password: User <code>tiger</code> as your password. Pathname: Open <code>c:\my_docs\projects</code> URL: Go to <code>http://www.oracle.com</code> User input: Enter <code>300</code> Username: Log on as <code>scott</code>
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (<i>but</i> Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do <i>not</i> save changes to the database. For further information, see <i>Oracle7 Server SQL Language Reference Manual</i> . Enter <code>user_id@us.oracle.com</code> , where <i>user_id</i> is the name of the user.
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select "Include a reusable module component" and click Finish. This subject is covered in Unit II, Lesson 3, "Working with Objects."
Uppercase	SQL column names, commands, functions, schemas, table names	Use the SELECT command to view information stored in the <code>LAST_NAME</code> column of the EMP table.

Convention	Element	Example
Arrow	Menu paths	Select File > Save.
Commas	Key sequences	Press and release keys one at a time: [Alternate], [F], [D]
Plus signs	Key combinations	Press and hold these keys simultaneously: [Ctrl]+[Alt]+[Del]

Typographic Conventions in Code

Convention	Element	Example
Caps and lowercase	Oracle Forms triggers	When-Validate-Item
Lowercase	Column names, table names	SELECT last_name FROM s_emp;
	Passwords	DROP USER scott IDENTIFIED BY tiger;
	PL/SQL objects	OG_ACTIVATE_LAYER (OG_GET_LAYER ('prod_pie_layer'))
Lowercase italic	Syntax variables	CREATE ROLE <i>role</i>
Uppercase	SQL commands and functions	SELECT userid FROM emp;

Introduction

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Course Objectives

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

- **Identify various components of the Oracle architecture**
- **Start up and shut down an Oracle database**
- **Create an operational database**
- **Manage Oracle control files, online redo log files, data files, tablespaces, segments, extents, and blocks**
- **Manage users, privileges, and resources**
- **Use Globalization Support features**

ORACLE

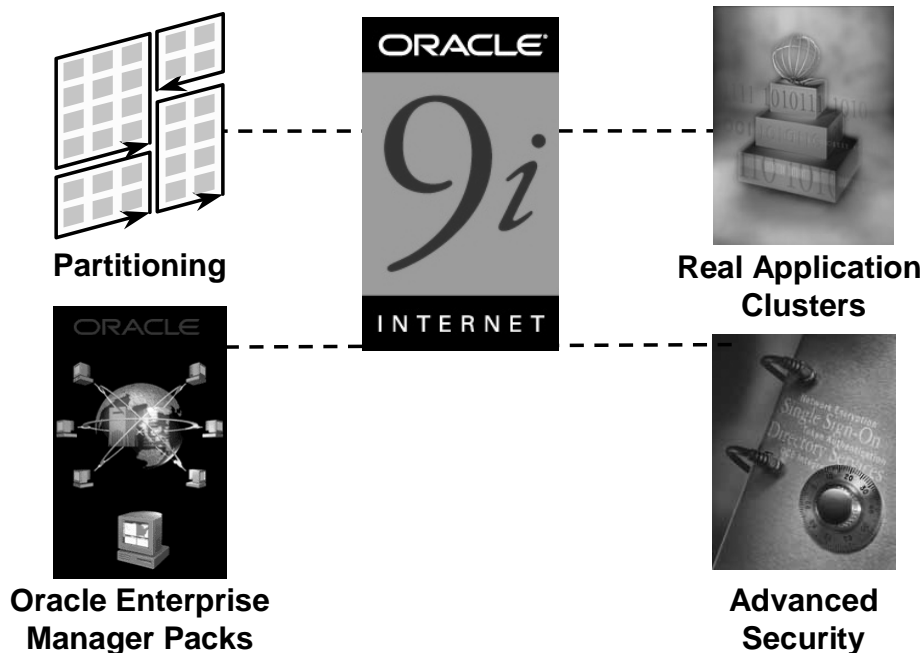
Copyright © Oracle Corporation, 2002. All rights reserved.

Course Objectives

This course is the first in a series of courses that cover the core database administrator tasks. The tasks covered in this course are:

- Outlining the Oracle architecture
- Planning and creating databases
- Managing the memory, process, physical, and logical structures
- Managing database users by controlling and monitoring their actions
- Using the Globalization Support features

Oracle9i Enterprise Edition



Copyright © Oracle Corporation, 2002. All rights reserved.

Oracle9i Enterprise Edition

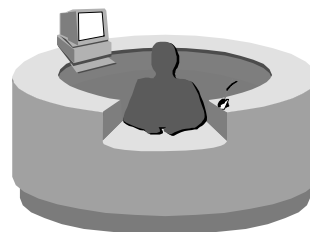
Oracle9i Enterprise Edition is an object-relational database that is scalable and easily manageable. The administration of the basic enterprise edition is discussed in this course. However, the following options provide additional functionality.

- **Partitioning:** Provides facilities for implementing large, scalable applications. Enables control over tables and indexes at a lower level of granularity than is possible with the basic enterprise edition.
- **Real Application Clusters:** Improves the scalability and availability of a database by allowing multiple instances of the Oracle software to access a single database.
- **Oracle Enterprise Manager Packs:** Built on top of the Oracle Enterprise Manager. Oracle Enterprise Manager Diagnostics, Tuning, and Change Management Packs are add-ons that provide DBAs with a set of tools for advanced diagnostics, monitoring, tuning, and change management of Oracle environments.
- **Advanced Security:** Provides client-server, server-server network security using encryption and data integrity checking, and supports enhanced user authentication services, using third-party security services.

Note: Options do require a license to be purchased.

Tasks of a Database Administrator

- To plan and create databases
- To manage database availability
- To manage physical and logical structures
- To manage storage based on design
- To manage security
- Network administration
- Backup and recovery
- Database tuning



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Tasks of a Database Administrator

Database administrators are responsible for maintaining the Oracle server so that the Oracle server can process user requests. An understanding of the Oracle architecture is necessary to maintain it effectively. This course will be focusing on outlining the Oracle architecture, and other administrator tasks such as: planning and creating databases, managing database availability, managing memory, physical and logical structures, and managing users and privileges.

Database administrator tasks covered in other courses:

The following tasks are discussed in other courses:

- Backup and recovery in *Oracle9i Database Administration Fundamentals II*
- Network administration in *Oracle9i Database Administration Fundamentals II*
- Database tuning in *Oracle9i Database Performance Tuning*

1

Oracle Architectural Components

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Outline the Oracle architecture and its main components**
- **List the structures involved in connecting a user to an Oracle instance**

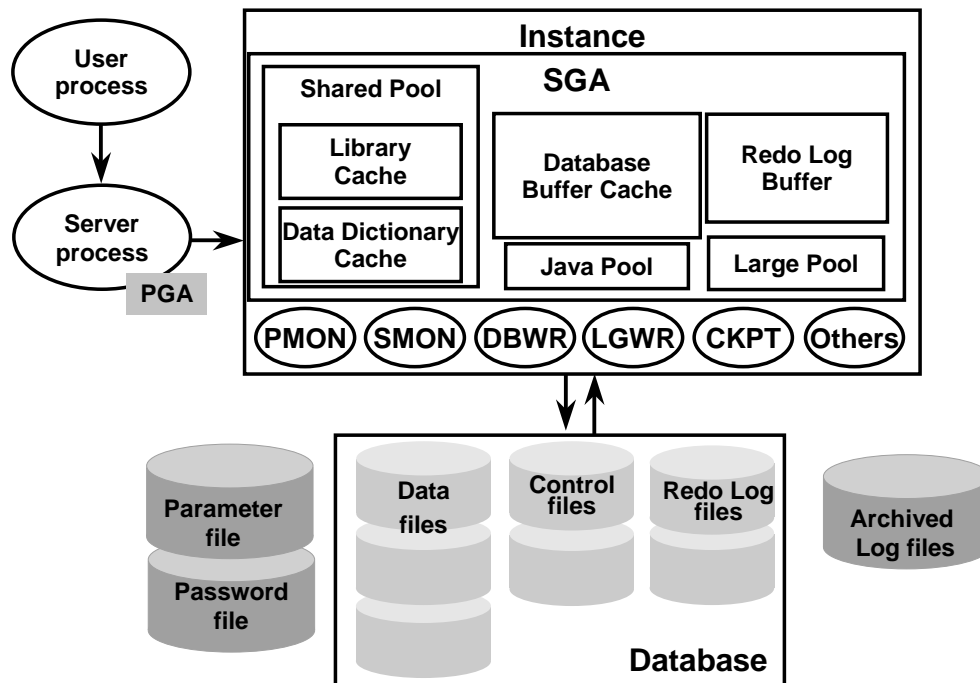
ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

This lesson introduces the Oracle server architecture by examining the physical, memory, process, and logical structures involved in establishing a database connection, creating a session, and executing SQL commands.

Overview of Primary Components



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Overview of Primary Components

The Oracle architecture includes a number of primary components, which are discussed further in this lesson.

- **Oracle server:** There are several files, processes, and memory structures in an Oracle server; however, not all of them are used when processing a SQL statement. Some are used to improve the performance of the database, to ensure that the database can be recovered in the event of a software or hardware error, or to perform other tasks necessary to maintain the database. **The Oracle server consists of an Oracle instance and an Oracle database.**
- **Oracle instance:** An Oracle instance is the combination of the background processes and memory structures. The instance must be started to access the data in the database. Every time an instance is started, a System Global Area (SGA) is allocated and Oracle background processes are started. Background processes perform functions on behalf of the invoking process. They consolidate functions that would otherwise be handled by multiple Oracle programs running for each user. The background processes perform input/output (I/O) and monitor other Oracle processes to provide increased parallelism for better performance and reliability.

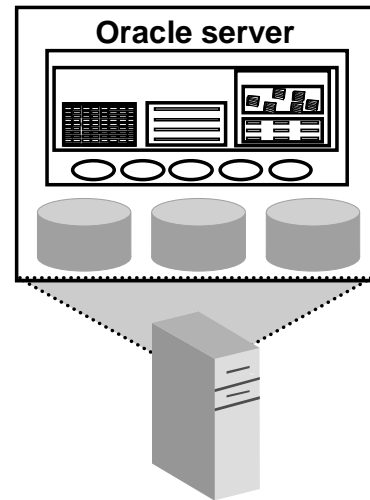
Overview of Primary Components (continued)

- **Oracle database:** An Oracle database consists of operating system files, also known as database files, that provide the actual physical storage for database information. The database files are used to ensure that the data is kept consistent and can be recovered in the event of a failure of the instance.
- **Other key files:** Nondatabase files are used to configure the instance, authenticate privileged users, and recover the database in the event of a disk failure.
- **User and server processes:** The user and server processes are the primary processes involved when a SQL statement is executed; however, other processes may help the server complete the processing of the SQL statement.
- **Other processes:** Many other processes exist that are used by other options, such as Advanced Queuing, Real Application Clusters, Shared Server, Advanced Replication, and so on. These processes are discussed within their respective courses.

Oracle Server

An Oracle server:

- Is a database management system that provides an open, comprehensive, integrated approach to information management
- Consists of an Oracle instance and an Oracle database



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

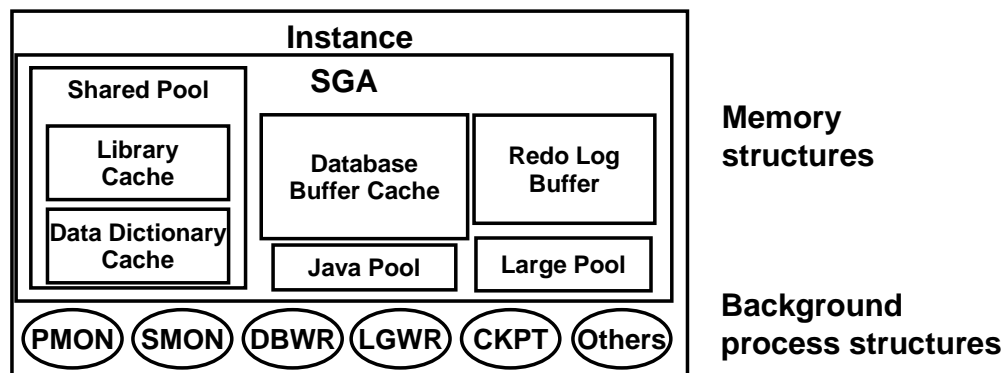
Oracle Server

The Oracle server is the key to information management. In general, an Oracle server must reliably manage a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this must be accomplished while delivering high performance. An Oracle server must also prevent unauthorized access and provide efficient solutions for failure recovery.

Oracle Instance

An Oracle instance:

- Is a means to access an Oracle database
- Always opens one and only one database
- Consists of memory and background process structures



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

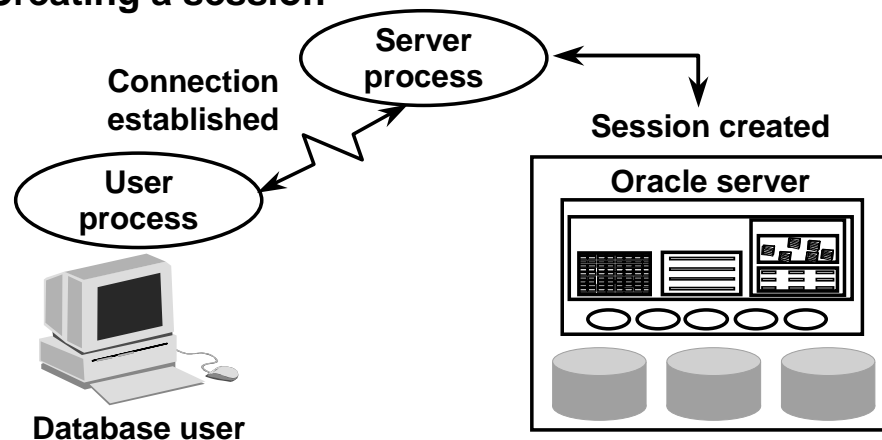
Oracle Instance

An Oracle instance consists of the System Global Area (SGA) memory structure and the background processes that are used to manage a database. An instance is identified by using methods specific to each operating system. The instance can open and use only one database at a time.

Establishing a Connection and Creating a Session

Connecting to an Oracle instance:

- Establishing a user connection
- Creating a session



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Establishing a Connection and Creating a Session

Before users can submit SQL statements to an Oracle database, they must connect to an instance.

- The user starts a tool such as SQL*Plus or runs an application developed using a tool such as Oracle Forms. This application or tool is executed as a user process.
- In the most basic configuration, when a user logs on to the Oracle server, a process is created on the computer running the Oracle server. This process is called a server process. The server process communicates with the Oracle instance on behalf of the user process that runs on the client. The server process executes SQL statements on behalf of the user.

Connection

A connection is a communication pathway between a user process and an Oracle server. A database user can connect to an Oracle server in one of three ways:

- The user logs on to the operating system running the Oracle instance and starts an application or tool that accesses the database on that system. The communication pathway is established using the interprocess communication mechanisms available on the host operating system.

Establishing a Connection and Creating a Session

Connection (continued)

- The user starts the application or tool on a local computer and connects over a network to the computer running the Oracle instance. In this configuration, called client-server, network software is used to communicate between the user and the Oracle server.
- In a three-tiered connection, the user's computer communicates over the network to an application or a network server, which is connected through a network to the machine running the Oracle instance. For example, the user runs a browser on a computer on a network to use an application residing on an NT server that retrieves data from an Oracle database running on a UNIX host.

Sessions

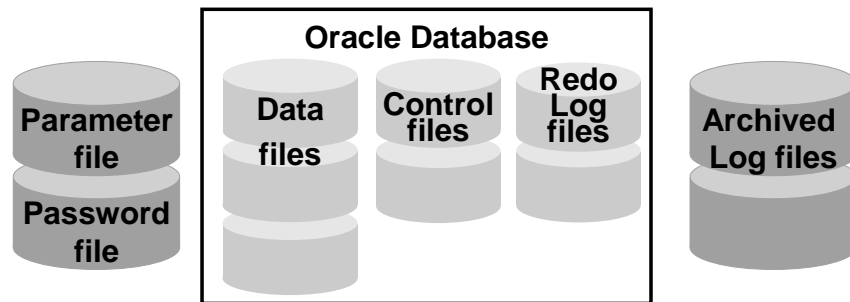
A session is a specific connection of a user to an Oracle server. The session starts when the user is validated by the Oracle server, and it ends when the user logs out or when there is an abnormal termination. For a given database user, many concurrent sessions are possible if the user logs on from many tools, applications, or terminals at the same time. Except for some specialized database administration tools, starting a database session requires that the Oracle server be available for use.

Note: The type of connection explained here, where there is a one-to-one correspondence between a user and server process, is called a dedicated server connection. When using a shared server configuration, it is possible for multiple user processes to share server processes.

Oracle Database

An Oracle database:

- Is a collection of data that is treated as a unit
- Consists of three file types



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Oracle Database

The general purpose of a database is to store and retrieve related information. An Oracle database has a logical and a physical structure. The physical structure of the database is the set of operating system files in the database. An Oracle database consists of three file types.

- Data files containing the actual data in the database
- Online redo log files containing a record of changes made to the database to enable recovery of the data in case of failures
- Control files containing information necessary to maintain and verify database integrity

Other Key File Structures

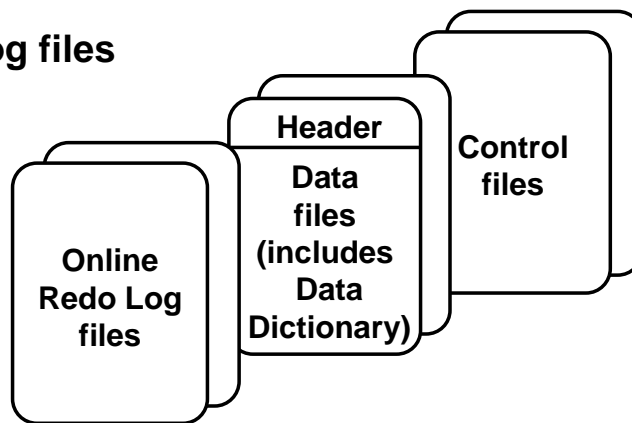
The Oracle server also uses other files that are not part of the database:

- The parameter file defines the characteristics of an Oracle instance. For example, it contains parameters that size some of the memory structures in the SGA.
- The password file authenticates users privileged to start up and shut down an Oracle instance.
- Archived redo log files are offline copies of the online redo log files that may be necessary to recover from media failures.

Physical Structure

The physical structure includes three types of files:

- Control files
- Data files
- Online redo log files



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Physical Structure

The physical structure of an Oracle database includes three types of files: control files, data files, and online redo log files.

Memory Structure

Oracle's memory structure consists of two memory areas known as:

- **System Global Area (SGA):** Allocated at instance start up, and is a fundamental component of an Oracle instance
- **Program Global Area (PGA):** Allocated when the server process is started

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

System Global Area

- **The SGA consists of several memory structures:**
 - Shared Pool
 - Database Buffer Cache
 - Redo Log Buffer
 - Other structures (for example, lock and latch management, statistical data)
- **There are two additional memory structures that can be configured within the SGA:**
 - Large Pool
 - Java Pool

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

System Global Area (SGA)

The SGA is also called the shared global area. It is used to store database information that is shared by database processes. It contains data and control information for the Oracle server and is allocated in the virtual memory of the computer where Oracle resides.

The following statement can be used to view SGA memory allocations:

```
SQL> SHOW SGA:
```

Total System Global Area	36437964	bytes
Fixed Size	6543794	bytes
Variable Size	19521536	bytes
Database Buffers	16777216	bytes
Redo Buffers	73728	bytes

System Global Area (continued)

Dynamic SGA

Beginning with Oracle9i, a dynamic SGA implements an infrastructure that allows the SGA configuration to change without shutting down the instance. This then allows the sizes of the Database Buffer Cache, Shared Pool, and Large Pool to be changed without shutting down the instance. Conceivably, the Database Buffer Cache and Shared Pool could be initially under configured and would grow and shrink depending upon their respective work loads, up to a maximum of `SGA_MAX_SIZE`.

Sizing the SGA

The size of the SGA is determined by several initialization parameters. The parameters that most affect SGA size are:

`DB_CACHE_SIZE`: The size of the cache of standard blocks. The default is 48 MB on UNIX and 52 MB on NT.

`LOG_BUFFER`: The number of bytes allocated for the Redo Log Buffer

`SHARED_POOL_SIZE`: The size in bytes of the area devoted to shared SQL and PL/SQL. The default is 16 MB. If 64 bit, then the default size is 64 MB.

`LARGE_POOL_SIZE`: The size of the Large Pool. The default is zero (Unless the `init.ora` parameter `PARALLEL_AUTOMATIC_TUNING` is set to `TRUE`, then the default is automatically calculated.)

`JAVA_POOL_SIZE`: The size of the Java Pool. The default is 24 MB.

Therefore, the size of the SGA can not exceed `SGA_MAX_SIZE` minus the values for `DB_CACHE_SIZE`, `LOG_BUFFER`, `SHARED_POOL_SIZE`, `LARGE_POOL_SIZE`, and `JAVA_POOL_SIZE`.

Note: Dynamic SGA and sizing are covered in further detail in the *Oracle9i Database Performance Tuning* course.

System Global Area

- **Is dynamic**
- **Sized by the `SGA_MAX_SIZE` parameter**
- **Allocated and tracked in granules by SGA components**
 - **Contiguous virtual memory allocation**
 - **Granule size based on total estimated `SGA_MAX_SIZE`**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

System Global Area (continued)

Unit of Allocation

A granule is a unit of contiguous virtual memory allocation. The size of a granule depends on the estimated total SGA size whose calculation is based on the value of the `SGA_MAX_SIZE` parameter.

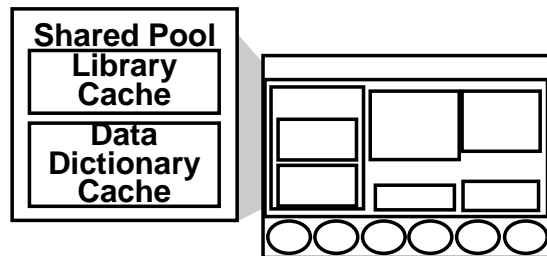
- 4 MB if estimated SGA size is < 128 MB
- 16 MB otherwise

The components (Database Buffer Cache, Shared Pool, and Large Pool) are allowed to grow and shrink based on granule boundaries. At instance start up, the Oracle server allocates granule entries, one for each granule to support `SGA_MAX_SIZE` bytes of address space. As start up continues, each component acquires as many granules as it requires. The minimum SGA configuration is three granules (one granule for fixed SGA [includes redo buffers]; one granule for Database Buffer Cache; one granule for Shared Pool).

Shared Pool

- **Used to store:**
 - Most recently executed SQL statements
 - Most recently used data definitions
- **It consists of two key performance-related memory structures:**
 - Library Cache
 - Data Dictionary Cache
- **Sized by the parameter**
SHARED_POOL_SIZE

```
ALTER SYSTEM SET  
SHARED_POOL_SIZE = 64M;
```



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Shared Pool

The Shared Pool environment contains both fixed and variable structures. The fixed structures remain relatively the same size, whereas the variable structures grow and shrink based on user and program requirements. The actual sizing for the fixed and variable structures is based on an initialization parameter and the work of an Oracle internal algorithm.

Sizing the Shared Pool

Because the Shared Pool is used for objects that can be shared globally, such as reusable SQL execution plans, PL/SQL packages, procedures, functions, and cursor information, it must be sized to accommodate the needs of both the fixed and variable areas. Memory allocation for the Shared Pool is determined by the SHARED_POOL_SIZE initialization parameter. It can be dynamically resized using ALTER SYSTEM SET. After performance analysis, this can be adjusted but the total SGA size cannot exceed SGA_MAX_SIZE.

Library Cache

- **Stores information about the most recently used SQL and PL/SQL statements**
- **Enables the sharing of commonly used statements**
- **Is managed by a least recently used (LRU) algorithm**
- **Consists of two structures:**
 - **Shared SQL area**
 - **Shared PL/SQL area**
- **Size determined by the Shared Pool sizing**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Library Cache

The Library Cache size is based on the sizing defined for the Shared Pool. Memory is allocated when a statement is parsed or a program unit is called. If the size of the Shared Pool is too small, statements are continually reloaded into the Library Cache, which affects performance. The Library Cache is managed by an LRU algorithm. As the cache fills, less recently used execution paths and parse trees are removed from the Library Cache to make room for the new entries. If the SQL or PL/SQL statements are not reused, they eventually are aged out.

The Library Cache consists of two structures:

- **Shared SQL:** The Shared SQL stores and shares the execution plan and parse tree for SQL statements run against the database. The second time that an identical SQL statement is run, it is able to take advantage of the parse information available in the shared SQL to expedite its execution. To ensure that SQL statements use a shared SQL area whenever possible, the text, schema, and bind variables must be exactly the same.
- **Shared PL/SQL:** The Shared PL/SQL area stores and shares the most recently executed PL/SQL statements. Parsed and compiled program units and procedures (functions, packages, and triggers) are stored in this area.

Data Dictionary Cache

- **A collection of the most recently used definitions in the database**
- **Includes information about database files, tables, indexes, columns, users, privileges, and other database objects**
- **During the parse phase, the server process looks at the data dictionary for information to resolve object names and validate access**
- **Caching data dictionary information into memory improves response time on queries and DML**
- **Size determined by the Shared Pool sizing**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Data Dictionary Cache

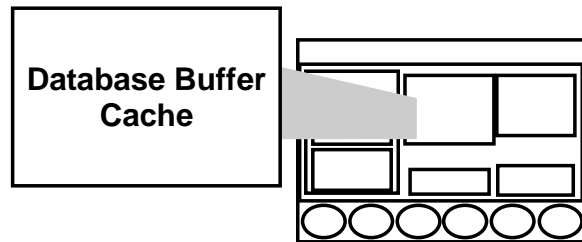
The Data Dictionary Cache is also referred to as the dictionary cache or row cache. Information about the database (user account data, data file names, segment names, extent locations, table descriptions, and user privileges) is stored in the data dictionary tables. When this information is needed by the server, the data dictionary tables are read, and the data that is returned is stored in the Data Dictionary Cache.

Sizing the Data Dictionary

The overall size is dependent on the size of the Shared Pool size and is managed internally by the database. If the Data Dictionary Cache is too small, then the database has to query the data dictionary tables repeatedly for information needed by the server. These queries are called recursive calls and are slower than the direct queries on the Data Dictionary Cache because direct queries do not use SQL.

Database Buffer Cache

- Stores copies of data blocks that have been retrieved from the data files
- Enables great performance gains when you obtain and update data
- Managed through an LRU algorithm
- `DB_BLOCK_SIZE` determines primary block size



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Database Buffer Cache

When a query is processed, the Oracle server process looks in the Database Buffer Cache for any blocks it needs. If the block is not found in the Database Buffer Cache, the server process reads the block from the datafile and places a copy in the Database Buffer Cache. Because subsequent requests for the same block may find the block in memory, the requests may not require physical reads. The Oracle server uses an LRU algorithm to age out buffers that have not been accessed recently to make room for new blocks in the Database Buffer Cache.

Database Buffer Cache

- **Consists of independent subcaches:**

- DB_CACHE_SIZE
- DB_KEEP_CACHE_SIZE
- DB_RECYCLE_CACHE_SIZE

- **Can be dynamically resized**

```
ALTER SYSTEM SET DB_CACHE_SIZE = 96M;
```

- **DB_CACHE_ADVICE set to gather statistics for predicting different cache size behavior**
- **Statistics displayed by V\$DB_CACHE_ADVICE**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Database Buffer Cache

Sizing the Database Buffer Cache

The size of each buffer in the Database Buffer Cache is equal to the size of an Oracle block, and it is specified by the DB_BLOCK_SIZE parameter. The Database Buffer Cache consists of independent subcaches for buffer pools and for multiple block sizes. The parameter DB_BLOCK_SIZE determines the primary block size, which is used for the SYSTEM tablespace.

Three parameters define the sizes of the Database Buffer Caches:

- DB_CACHE_SIZE: Sizes the default buffer cache only; it always exists and cannot be set to zero
- DB_KEEP_CACHE_SIZE: Sizes the keep buffer cache, which is used to retain blocks in memory that are likely to be reused
- DB_RECYCLE_CACHE_SIZE: Sizes the recycle buffer cache, which is used to eliminate blocks from memory that have little chance of being reused

Database Buffer Cache (continued)

Buffer Cache Advisory

The Buffer Cache Advisory feature enables and disables statistics gathering for predicting behavior with different cache sizes. The information provided by these statistics can help you size the Database Buffer Cache optimally for a given workload. The Buffer Cache Advisory information is collected and displayed through the `V$DB_CACHE_ADVICE` view.

The Buffer Cache Advisory is enabled via the `DB_CACHE_ADVICE` initialization parameter. It is a dynamic parameter, and can be altered using `ALTER SYSTEM`. Three values (OFF, ON, READY) are available.

DB_CACHE_ADVICE Parameter Values

OFF: Advisory is turned off and the memory for the advisory is not allocated.

ON: Advisory is turned on and both cpu and memory overhead is incurred.

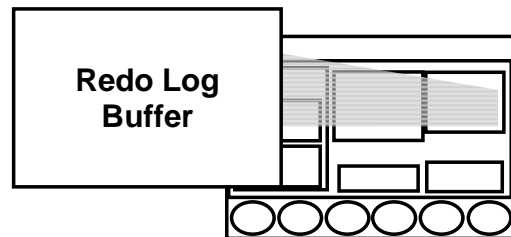
Attempting to set the parameter to the ON state when it is in the OFF state may lead to the following error: `ORA-4031 Inability to allocate from the Shared Pool when the parameter is switched to ON`. If the parameter is in a READY state it can be set to ON without error because the memory is already allocated.

READY: Advisory is turned off but the memory for the advisory remains allocated.

Allocating the memory before the advisory is actually turned on will avoid the risk of `ORA-4031`. If the parameter is switched to this state from OFF, it is possible that an `ORA-4031` will be raised.

Redo Log Buffer

- Records all changes made to the database data blocks
- Primary purpose is recovery
- Changes recorded within are called redo entries
- Redo entries contain information to reconstruct or redo changes
- Size defined by LOG_BUFFER



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Redo Log Buffer

The Redo Log Buffer is a circular buffer that contains changes made to datafile blocks. This information is stored in redo entries. Redo entries contain the information necessary to re-create the data prior to the change made by INSERT, UPDATE, DELETE, CREATE, ALTER, or DROP operations.

Sizing the Redo Log Buffer

The size of the Redo Log Buffer is defined by the LOG_BUFFER initialization parameter.

Note: Sizing the Redo Log Buffer is covered in further detail in the *Oracle9i Database Performance Tuning* course. Refer to the “Managing Redo Log Files” lesson for details regarding online redo log files.

Large Pool

- **An optional area of memory in the SGA**
- **Relieves the burden placed on the Shared Pool**
- **Used for:**
 - **Session memory (UGA) for the Shared Server**
 - **I/O server processes**
 - **Backup and restore operations or RMAN**
 - **Parallel execution message buffers**
- **PARALLEL_AUTOMATIC_TUNING set to TRUE**
- **Does not use an LRU list**
- **Sized by LARGE_POOL_SIZE**
- **Can be dynamically resized**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Large Pool

By allocating session memory from the Large Pool for Shared Server, Oracle XA, or parallel query buffers, Oracle can use the Shared Pool primarily for caching Shared SQL statements. Thus relieving the burden on areas within the Shared Pool. The Shared Pool does not have to give up memory for caching SQL parse trees in favor of Shared Server session information, I/O, and backup and recover processes. The performance gain is from the reduction of overhead from increasing and shrinkage of the shared SQL cache.

Backup and Restore

Recovery Manager (RMAN) uses the Large Pool when the `BACKUP_DISK_IO=n` and `BACKUP_TAPE_IO_SLAVE=TRUE` parameters are set. If the Large Pool is configured but is not large enough, the allocation of memory from the Large Pool fails. RMAN writes an error message to the alert log file and does not use I/O slaves for backup or restore.

Parallel Execution

The Large Pool is used if `PARALLEL_AUTOMATIC_TUNING` is set to `TRUE`, otherwise, these buffers are allocated to the Shared Pool.

Large Pool (continued)

Sizing the Large Pool

The Large Pool is sized in bytes defined by the `LARGE_POOL_SIZE` parameter. This parameter can be dynamically resized using the `ALTER SYSTEM SET` command.

```
SQL> ALTER SYSTEM SET LARGE_POOL_SIZE=24MB
```

Large Pool and LRU lists

The Large Pool does not have an LRU list. It is different from reserved space in the Shared Pool, which uses an LRU list.

Java Pool

- **Services parsing requirements for Java commands**
- **Required if installing and using Java**
- **Sized by `JAVA_POOL_SIZE` parameter**

ORACLE

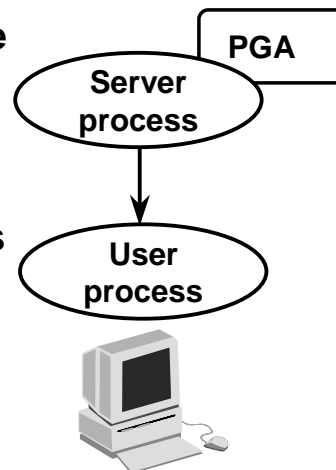
Copyright © Oracle Corporation, 2002. All rights reserved.

Java Pool

The Java Pool is an optional setting but is required if you are installing and using Java. Its size is set, in bytes, using the `JAVA_POOL_SIZE` parameter. In Oracle9i, the default size of the Java Pool is 24 MB.

Program Global Area

- **Memory reserved for each user process connecting to an Oracle database**
- **Allocated when a process is created**
- **Deallocated when the process is terminated**
- **Used by only one process**



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Program Global Area (PGA)

The Program Global Area or Process Global Area (PGA) is a memory region that contains data and control information for a single server process or a single background process. The PGA is allocated when a process is created and deallocated when the process is terminated. In contrast to the SGA, which is shared by several processes, the PGA is an area that is used by only one process.

Contents of PGA

The contents of the PGA memory varies, depending whether the instance is running in a dedicated server or shared server configuration. Generally the PGA memory includes these components:

- **Private SQL Area:** Contains data such as bind information and run-time memory structures. Each session that issues a SQL statement has a private SQL area. Each user that submits the same SQL statement has his or her own private SQL area that uses a single shared SQL area. Thus, many private SQL areas can be associated with the same shared SQL area. The private SQL area of a cursor is divided into two areas:



Program Global Area (continued)

- **Persistent area:** Contains bind information, and is freed only when the cursor is closed
- **Run-time area:** Created as the first step of an execute request. For INSERT, UPDATE, and DELETE commands, this area is freed after the statement has been executed. For queries, this area is freed only after all rows are fetched or the query is canceled.

The location of the private SQL area depends on the type of connection established for the session. In a dedicated server environment, the private SQL areas are located in the PGA of their server process. In a shared server environment, the private SQL areas are located in the SGA.

The management of private SQL areas is the responsibility of the user process. The number of private SQL areas that a user process can allocate is always limited by the `OPEN_CURSORS` initialization parameter. The default value of this parameter is 50.

- **Session Memory:** Consists of memory allocated to hold a session's variables and other information related to the session. For a shared server environment, the session memory is shared and not private.
- **SQL Work Areas:** Used for memory-intensive operations such as: Sort, Hash-join, Bitmap merge, Bitmap create. The size of a work area can be controlled and tuned. Beginning with Oracle9i, the size of the work area can be automatically and globally managed. This is enabled by setting the `WORKAREA_SIZE_POLICY` parameter to `AUTO`, which is the default, and the `PGA_AGGREGATE_TARGET` initialization parameter. The `PGA_AGGREGATE_TARGET` parameter is set to specify the target aggregate amount of PGA memory available to the instance. This parameter is only a target and can be dynamically modified at the instance level. It will accept a number of bytes, kilobytes, megabytes, or gigabytes. When these parameters are set, sizing of work areas becomes automatic and all `*_AREA_SIZE`.

Prior to Oracle9i, the DBA controlled the maximum size of SQL work areas by setting the following parameters: `SORT_AREA_SIZE`, `HASH_AREA_SIZE`, `BITMAP_MERGE_AREA_SIZE`, and `CREATE_BITMAP_AREA_SIZE`. Setting these parameters can be difficult because the maximum work area size is ideally selected on the basis of the data input size and the total number of work areas active in the system. These two factors vary considerably from one work area to another and from one time to another.

Program Global Area (continued)

Differences in Memory Allocation Between Dedicated Server and Shared Server

The content of the PGA memory varies, depending whether the instance is running in a dedicated server or shared server configuration. Generally the PGA memory includes the following components:

Memory Area	Dedicated Server	Shared Server
Nature of Session Memory	Private	Shared
Location of Persistent Area	PGA	SGA
Location of Run-time Area (SELECT)	PGA	SGA
Location of Run-time Area (DML/DDL)	PGA	PGA

Process Structure

Oracle takes advantage of various types of processes:

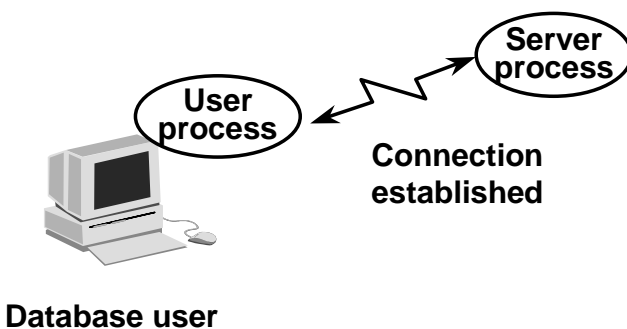
- **User process:** Started at the time a database user requests connection to the Oracle server
- **Server process:** Connects to the Oracle instance and is started when a user establishes a session
- **Background processes:** Started when an Oracle instance is started

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

User Process

- A program that requests interaction with the Oracle server
- Must first establish a connection
- Does not interact directly with the Oracle server



ORACLE

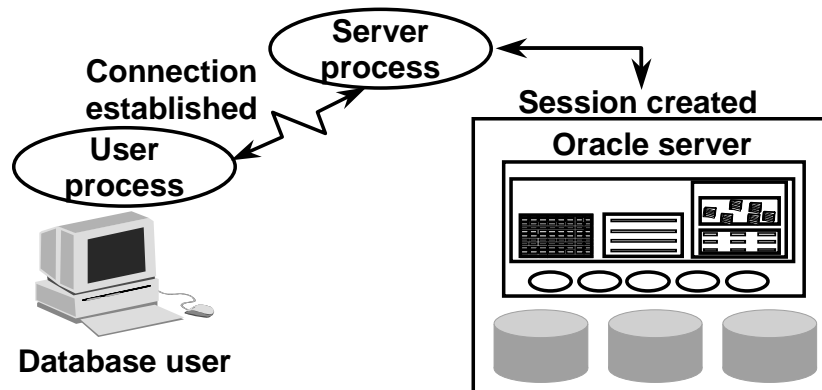
Copyright © Oracle Corporation, 2002. All rights reserved.

User Process

A database user who needs to request information from the database must first make a connection with the Oracle server. The connection is requested using a database interface tool, such as SQL*Plus, and beginning the user process. The user process does not interact directly with the Oracle server. Rather it generates calls through the user program interface (UPI), which creates a session and starts a server process.

Server Process

- A program that directly interacts with the Oracle server
- Fulfills calls generated and returns results
- Can be dedicated or shared server



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Server Process

Once a user has established a connection, a server process is started to handle the user processes requests. A server process can be either a dedicated server process or a shared server process. In a dedicated server environment, the server process handles the request of a single user process. Once a user process disconnects, the server process is terminated. In a shared server environment, the server process handles the request of several user processes. The server process communicates with the Oracle server using the Oracle Program Interface (OPI).

Note: Allocation of server process in a dedicated environment versus a shared environment is covered in further detail in the *Oracle9i Database Performance Tuning* course.

Background Processes

Maintains and enforces relationships between physical and memory structures:

- **Mandatory background processes:**

DBWn PMON CKPT
LGWR SMON

- **Optional background processes:**

ARCn LMDn QMNn
CJQ0 LMON RECO
Dnnn LMS Snnn
LCKn Pnnn

ORACLE

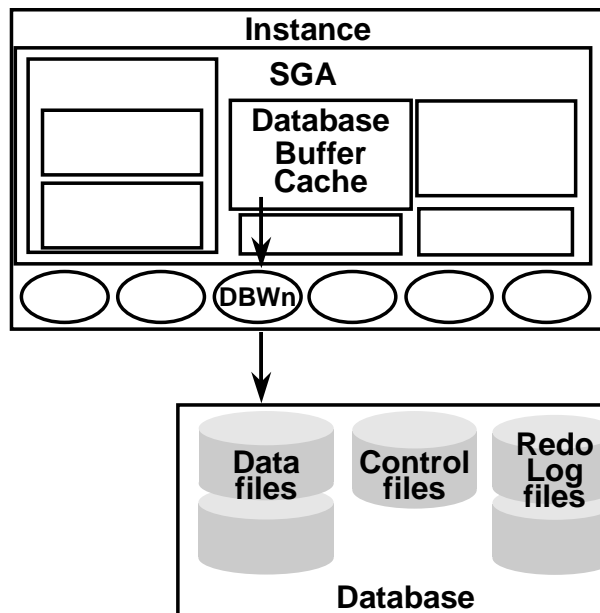
Copyright © Oracle Corporation, 2002. All rights reserved.

Background Processes

The Oracle architecture has five mandatory background processes that are discussed further in this lesson. In addition to the mandatory list, Oracle has many optional background process that are started when their option is being used. These optional processes are not within the scope of this course, with the exception of the background process, ARCn. Following is a list of some optional background processes:

- ARCn: Archiver
- CJQ0: Coordinator Job Queue background process
- Dnnn: Dispatcher
- LCKn: RAC Lock Manager–Instance Locks
- LMDn: RAC DLM Monitor–Remote Locks
- LMON: RAC DLM Monitor–Global Locks
- LMS: RAC Global Cache Service
- Pnnn: Parallel Query Slaves
- QMNn: Advanced Queuing
- RECO: Recoverer
- Snnn: Shared Server

Database Writer (DBWn)



DBWn writes when:

- Checkpoint occurs
- Dirty buffers reach threshold
- There are no free buffers
- Timeout occurs
- RAC ping request is made
- Tablespace OFFLINE
- Tablespace READ ONLY
- Table DROP or TRUNCATE
- Tablespace BEGIN BACKUP

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

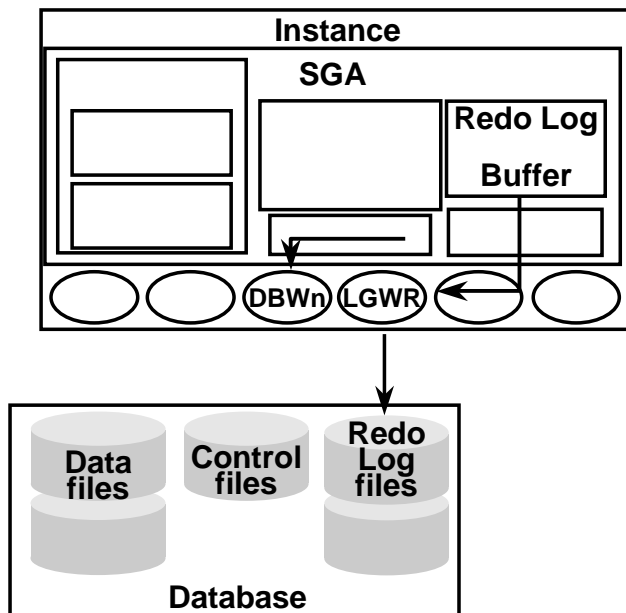
Database Writer (DBWn)

The server process records changes to undo and data blocks in the Database Buffer Cache. DBWn writes the dirty buffers from the Database Buffer Cache to the data files. It ensures that a sufficient number of free buffers (buffers that can be overwritten when server processes need to read in blocks from the data files) are available in the Database Buffer Cache. Database performance is improved because server processes make changes only in the Database Buffer Cache.

DBWn defers writing to the data files until one of the following events occurs:

- Incremental or normal checkpoint
- The number of dirty buffers reaches a threshold value
- A process scans a specified number of blocks when scanning for free buffers and cannot find any
- Timeout occurs
- A ping request in Real Application Clusters (RAC) environment
- Placing a normal or temporary tablespace offline
- Placing a tablespace in read-only mode
- Dropping or truncating a table
- ALTER TABLESPACE tablespace name BEGIN BACKUP

Log Writer (LGWR)



LGWR writes:

- At commit
- When one-third full
- When there is 1 MB of redo
- Every three seconds
- Before DBWn writes

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Log Writer (LGWR)

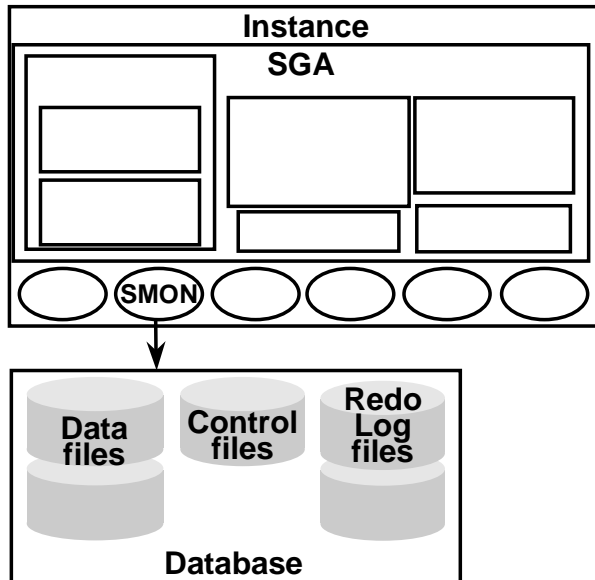
LGWR performs sequential writes from the Redo Log Buffer to the online redo log file under the following situations:

- When a transaction commits
- When the Redo Log Buffer is one-third full
- When there is more than 1 MB of changes recorded in the Redo Log Buffer
- Before DBWn writes modified blocks in the Database Buffer Cache to the data files
- Every three seconds

Because the redo is needed for recovery, LGWR confirms the commit operation only after the redo is written to disk.

LGWR can also call on DBWn to write to the data files.

System Monitor (SMON)



Responsibilities:

- **Instance recovery**
 - Rolls forward changes in online redo log files
 - Opens database for user access
 - Rolls back uncommitted transactions
- **Coalesces free space**
- **Deallocates temporary segments**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

System Monitor (SMON)

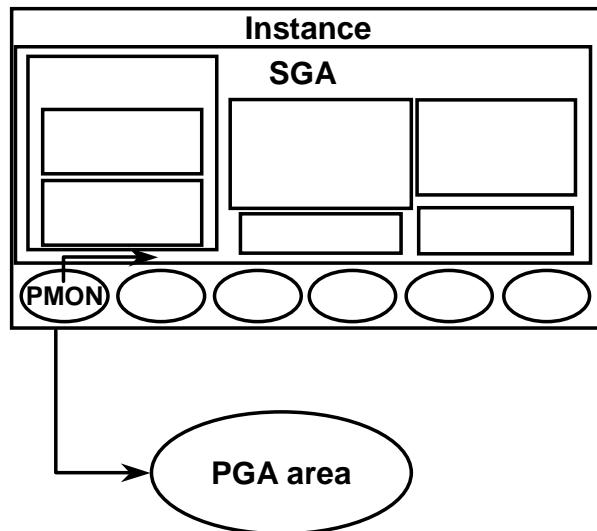
If the Oracle instance fails, any information in the SGA that has not been written to disk is lost. For example, the failure of the operating system causes an instance failure. After the loss of the instance, the background process SMON automatically performs instance recovery when the database is reopened. Instance recovery consists of the following steps:

1. Rolling forward to recover data that has not been recorded in the data files but that has been recorded in the online redo log file. This data has not been written to disk because of the loss of the SGA during instance failure. During this process, SMON reads the online redo log files and applies the changes recorded in the online redo log file to the data blocks. Because all committed transactions have been written to the online redo log files, this process completely recovers these transactions.
2. Opening the database so that users can log on. Any data that is not locked by unrecovered transactions is immediately available.
3. Rolling back uncommitted transactions. They are rolled back by SMON or by the individual server processes as they access locked data.

SMON also performs some space maintenance functions:

- It combines, or coalesces, adjacent areas of free space in the data files.
- It deallocates temporary segments to return them as free space in data files.

Process Monitor (PMON)



Cleans up after failed processes by:

- Rolling back the transaction
- Releasing locks
- Releasing other resources
- Restarting dead dispatchers

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

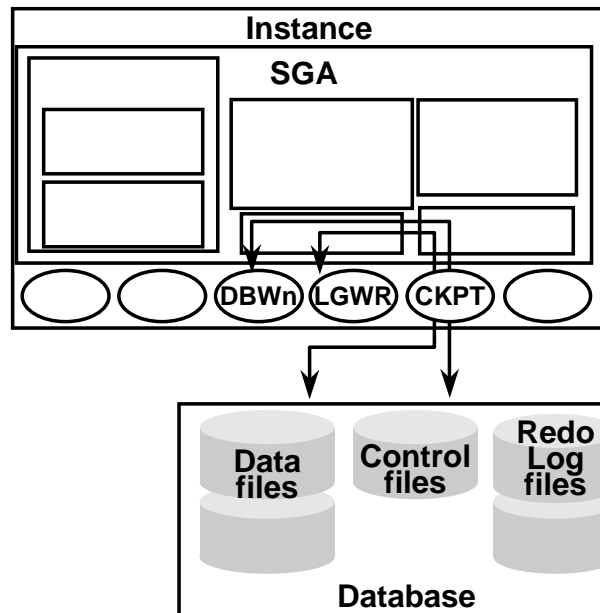
Process Monitor (PMON)

The background process PMON cleans up after failed processes by:

- Rolling back the user's current transaction
- Releasing all currently held table or row locks
- Freeing other resources currently reserved by the user
- Restarts dead dispatchers

Dispatchers are covered in further detail in the *Oracle9i Database Administration Fundamentals II* course.

Checkpoint (CKPT)



Responsible for:

- Signaling DBWn at checkpoints
- Updating datafile headers with checkpoint information
- Updating control files with checkpoint information

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Checkpoint (CKPT)

Every three seconds the CKPT process stores data in the control file to identify that place in the online redo log file where recovery is to begin, which is called a checkpoint. The purpose of a checkpoint is to ensure that all of the buffers in the Database Buffer Cache that were modified prior to a point in time have been written to the data files. This point in time (called the checkpoint position) is where database recovery is to begin in the event of an instance failure. DBWn will already have written all of the buffers in the Database Buffer Cache that were modified prior to that point in time. Prior to Oracle9i, this was done at the end of the online redo log file. In the event of a log switch CKPT also writes this checkpoint information to the headers of the data files.

Checkpoints are initiated for the following reasons:

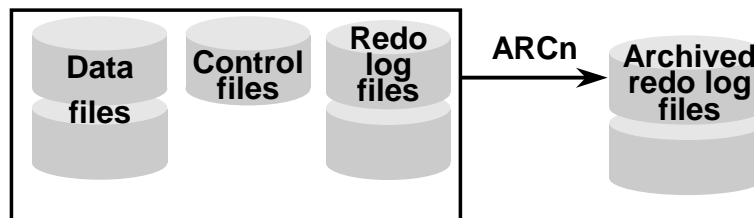
- To ensure that modified data blocks in memory are written to disk regularly so that data is not lost in case of a system or database failure.
- To reduce the time required for instance recovery. Only the online redo log file entries following the last checkpoint need to be processed for recovery to occur.
- To ensure that all committed data has been written to the data files during shut down.

Checkpoint information written by CKPT includes checkpoint position, system change number, location in the online redo log file to begin recovery, information about logs, and so on.

Note: CKPT does not write data blocks to disk or redo blocks to the online redo log files.

Archiver (ARCn)

- **Optional background process**
- **Automatically archives online redo log files when ARCHIVELOG mode is set**
- **Preserves the record of all changes made to the database**



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Archiver (ARCn)

ARCn is an optional background process, however, it is crucial to recovering a database after the loss of a disk. As online redo log files get filled, the Oracle server begins writing to the next online redo log file. The process of switching from one online redo log file to another is called a log switch. The ARCn process initiates backing up, or archiving, of the filled log group at every log switch. It automatically archives the online redo log file before the log can be reused, so all of the changes made to the database are preserved. This enables recovery of the database to the point of failure even if a disk drive is damaged.

Archiving Online Redo Log Files

One of the important decisions that a DBA has to make is whether to configure the database to operate in ARCHIVELOG or in NOARCHIVELOG mode.

NOARCHIVELOG mode: In NOARCHIVELOG mode, the online redo log files are overwritten each time a log switch occurs. LGWR does not overwrite an online redo log file group until the checkpoint for that group is complete. This ensures that committed data can be recovered if there is an instance crash. During the instance crash, only the SGA is lost. There is no loss of disks, only memory. For example, an operating system crash causes an instance crash.

Archiver (ARCn) (continued)

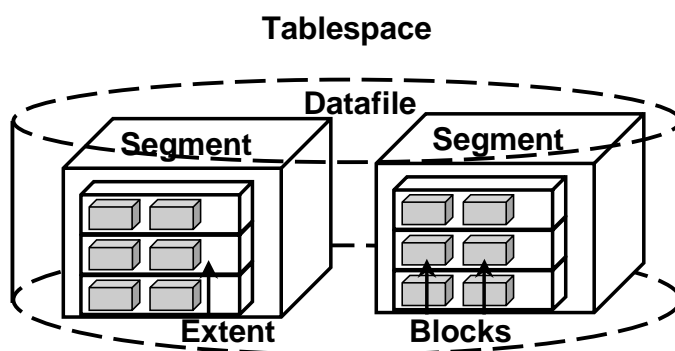
Archiving Online Redo Log Files (continued)

ARCHIVELOG mode: If the database is configured to run in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived before they can be used again. Because changes made to the database are recorded in the online redo log files, the database administrator can use the physical backup of the data files and the archived redo log files to recover the database without losing any committed data because of any single point of failure, including the loss of a disk. Usually, a production database is configured to run in ARCHIVELOG mode.

Archive log modes are covered in further detail in the *Oracle9i Database Administration Fundamentals II* course.

Logical Structure

- Dictates how the physical space of a database is used
- Hierarchy consisting of tablespaces, segments, extents, and blocks



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Logical Structure

A logical structure hierarchy exists as follows:

- An Oracle database contains at least one tablespace.
- A tablespace contains one or more segments.
- A segment is made up of extents.
- An extent is made up of logical blocks.
- A block is the smallest unit for read and write operations.

The Oracle database architecture includes logical and physical structures that make up the database.

- The physical structure includes the control files, online redo log files, and data files that make up the database.
- The logical structure includes tablespaces, segments, extents, and data blocks.

The Oracle server enables fine-grained control of disk space use through tablespace and logical storage structures, including segments, extents, and data blocks.

Logical Structure (continued)

Tablespaces

The data in an Oracle database is stored in tablespaces.

- An Oracle database can be logically grouped into smaller logical areas of space known as tablespaces.
- A tablespace can belong to only one database at a time.
- Each tablespace consists of one or more operating system files, which are called data files.
- A tablespace may contain one or more segments.
- Tablespaces can be brought online while the database is running.
- Except for the SYSTEM tablespace or a tablespace with an active undo segment, tablespaces can be taken offline, leaving the database running.
- Tablespaces can be switched between read/write and read-only status.

Data Files (Not a logical structure)

- Each tablespace in an Oracle database consists of one or more files called data files. These are physical structures that conform with the operating system on which the Oracle server is running.
- A data file can belong to only one tablespace.
- An Oracle server creates a data file for a tablespace by allocating the specified amount of disk space plus a small amount of overhead.
- The database administrator can change the size of a data file after its creation or can specify that a data file should dynamically grow as objects in the tablespace grow.

Segments

- A segment is the space allocated for a specific logical storage structure within a tablespace.
- A tablespace may consist of one or more segments.
- A segment cannot span tablespaces; however, a segment can span multiple data files that belong to the same tablespace.
- Each segment is made up of one or more extents.

Extents

Space is allocated to a segment by extents.

- One or more extents make up a segment.
 - When a segment is created, it consists of at least one extent.
 - As the segment grows, extents are added to the segment.
 - The DBA can manually add extents to a segment.
- An extent is a set of contiguous Oracle blocks.
- An extent cannot span data files, and therefore, it must exist in one datafile.

Logical Structure (continued)

Data Blocks

The Oracle server manages the storage space in the data files in units called Oracle blocks or data blocks.

- At the finest level of granularity, the data in an Oracle database is stored in data blocks.
- Oracle data blocks are the smallest units of storage that the Oracle server can allocate, read, or write.
- One data block corresponds to one or more operating system blocks allocated from an existing data file.
- The standard data block size for an Oracle database is specified by the `DB_BLOCK_SIZE` initialization parameter when the database is created.
- The data block size should be a multiple of the operating system block size to avoid unnecessary I/O.
- The maximum data block size is dependent on the operating system.

Processing SQL Statements

- **Connect to an instance using:**
 - User process
 - Server process
- **The Oracle server components that are used depend on the type of SQL statement:**
 - Queries return rows
 - DML statements log changes
 - Commit ensures transaction recovery
- **Some Oracle server components do not participate in SQL statement processing.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Processing SQL Statements

Processing a Query

- **Parse:**
 - Search for identical statement
 - Check syntax, object names, and privileges
 - Lock objects used during parse
 - Create and store execution plan
- **Bind:** Obtain values for variables
- **Execute:** Process statement
- **Fetch:** Return rows to user process

Processing SQL Statements (continued)

Processing a DML Statement

- **Parse:** Same as the parse phase used for processing a query
- **Bind:** Same as the bind phase used for processing a query
- **Execute:**
 - If the data and undo blocks are not already in the Database Buffer Cache, the server process reads them from the data files into the Database Buffer Cache.
 - The server process places locks on the rows that are to be modified. The undo block is used to store the before image of the data, so that the DML statements can be rolled back if necessary.
 - The data blocks record the new values of the data.
 - The server process records the before image to the undo block and updates the data block. Both of these changes are made in the Database Buffer Cache. Any changed blocks in the Database Buffer Cache are marked as dirty buffers. That is, buffers that are not the same as the corresponding blocks on the disk.
 - The processing of a DELETE or INSERT command uses similar steps. The before image for a DELETE contains the column values in the deleted row, and the before image of an INSERT contains the row location information.

Processing a DDL Statement

The execution of DDL (data definition language) statements differs from the execution of DML (data manipulation language) statements and queries, because the success of a DDL statement requires write access to the data dictionary. For these statements, parsing actually includes parsing, data dictionary lookup, and execution. Transaction management, session management, and system management SQL statements are processed using the parse and execute stages. To re-execute them, simply perform another execute.

Summary

In this lesson, you should have learned how to:

- **Explain database files: data files, control files, online redo log files**
- **Explain SGA memory structures: Database Buffer Cache, Shared Pool, and Redo Log Buffer**
- **Explain primary background processes: DBWn, LGWR, CKPT, PMON, SMON**
- **Explain the use of the optional background process ARCn**
- **Identify optional and conditional background processes**
- **Explain logical hierarchy**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 1 Overview

This practice covers the following topics:

- **Reviewing architectural components**
- **Identifying structures involved in connecting a user to an Oracle instance**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 1: Oracle Architectural Components

- 1 Which one of the following statements is true?
 - a An Oracle server is a collection of data consisting of three file types.
 - b A user establishes a connection with the database by starting an Oracle instance.
 - c A connection is a communication pathway between the Oracle server and the Oracle instance.
 - d A session starts when a user is validated by the Oracle server.
- 2 Which one of the following memory areas is not part of the SGA?
 - a Database Buffer Cache
 - b PGA
 - c Redo Log Buffer
 - d Shared Pool
- 3 Which two of the following statements are true about the Shared Pool?
 - a The Shared Pool consists of the Library Cache, Data Dictionary Cache, Shared SQL area, Java Pool, and Large Pool.
 - b The Shared Pool is used to store the most recently executed SQL statements.
 - c The Shared Pool is used for an object that can be shared globally.
 - d The Library Cache consists of the Shared SQL and Shared PL/SQL areas.
- 4 Which one of the following memory areas is used to cache the data dictionary information?
 - a Database Buffer Cache
 - b PGA
 - c Redo Log Buffer
 - d Shared Pool
- 5 The primary purpose of the Redo Log Buffer is to record all changes to the database data blocks.
 - a True
 - b False
- 6 The PGA is a memory region that contains data and control information for multiple server processes or multiple background processes.
 - a True
 - b False
- 7 Which of the following becomes available when an Oracle instance is started?
 - a User process
 - b Server process
 - c Background processes
- 8 Identify five mandatory background processes.

Practice 1: Oracle Architectural Components (continued)

- 9** Match the process with its task.
- | | |
|--------------------------|---|
| a Database Writer | 1 Assists with writing to the data file headers |
| b Log Writer | 2 Is responsible for instance recovery |
| c System Monitor | 3 Cleans up after failed processes |
| d Process Monitor | 4 Records database changes for recovery purposes |
| e Checkpoint | 5 Writes dirty buffers to the data files |
- 10** The physical structure of an Oracle database consists of control files, data files, and online redo log files.
- a** True
 - b** False
- 11** Place the following structures in order of hierarchy beginning with database.
- a** Tablespaces
 - b** Extent
 - c** Segment
 - d** Database
 - e** Block
- 12** Identify the components of an Oracle server.
-
-
- 13** Identify the components of an Oracle instance.
-
-
- 14** Identify three file types that make up an Oracle database.
-
-
-

2

Getting Started with the Oracle Server

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Identify common database administration tools available to a DBA**
- **Identify the features of the Oracle Universal Installer**
- **Use SQL*Plus to interact and manipulate an Oracle database**
- **List the main components of Oracle Enterprise Manager**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Database Administration Tools

Tool	Description
Oracle Universal Installer (OUI)	Used to install, upgrade, or remove software components
Oracle Database Configuration Assistant	A graphical user interface tool that interacts with the OUI, or can be used independently, to create, delete, or modify a database
SQL*Plus	A utility to access data in an Oracle database
Oracle Enterprise Manager	A graphical interface used to administer, monitor, and tune one or more databases

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Database Administration Tools

The tools listed are covered in this course, but they are only a subset of the utilities supplied by Oracle.

Oracle Universal Installer

- **Used to install, upgrade, or remove software components, and create a database**
- **Based on a Java engine**
- **Features include**
 - **Automatic dependency resolution**
 - **Allows for Web-based installations**
 - **Tracking inventory of component and suite installations**
 - **Deinstallation of installed components**
 - **Support for multiple Oracle homes**
 - **Support for globalization technology**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Oracle Universal Installer

The Java-based Oracle Universal Installer offers an installation solution for all Java-enabled platforms, allowing for a common installation flow and user experience independent of the platform.

The Universal Installer

- Detects dependencies among components and performs an installation accordingly
- Can be used to point to a URL where a release or staging area was defined and install software remotely over HTTP
- Can be used to remove products installed. The deinstallation actions are the “undo” of installation actions.
- Maintains an inventory of all the Oracle homes on a target machine, their names, products, and the versions of the products installed on them
- Detects the language of the operating system and runs the installation session in that language
- Can be run in interactive mode or silent mode. Oracle Universal Installer is run in silent (or non-interactive) mode using a response file.

Starting the Universal Installer

- To start Oracle Universal Installer on UNIX:

```
$ ./runInstaller
```

- To start Oracle Universal Installer on NT:

```
Start > Programs > Oracle Installation  
Products > Universal Installer
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Starting the Universal Installer

UNIX

The installation program is called `runInstaller` and is located in the `oracle\oui\install` directory.

On UNIX do not run the Installer as a root user.

NT

The installation program is called `setup.exe` and is located in the `Program Files/Oracle/oui/install` directory.

Note: See your operating system–specific Oracle documentation for information about installing Oracle Server on your platform.

Non-Interactive Installation Using Response Files

- **Allows for no user interaction**
- **Response files:**
 - **Templates must be edited.**
 - **Text files contain variables and values.**
 - **Parameters are customized.**
- **To start Universal Installer in non-interactive mode:**

```
./runInstaller -responsefile myrespfile -silent
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Non-Interactive Installation Using Response Files

Non-interactive installation is performed when no user action is intended or if nongraphical terminals are used for installation.

Installation parameters are customized using a response file. A response file is a text file containing variables and values used by Oracle Universal Installer during the installation process. Example of installation parameters include values for ORACLE_HOME and installation types (that is, Typical or Custom Install).

The user first must copy and edit the response file to specify the components to install.

UNIX

Response file templates are available in the `stage/response` directory. On UNIX systems, enter the following at the command line in the directory where the Universal Installer is installed:

```
./runInstaller -responsefile filename [-silent] [-nowelcome]
```

NT

Response file templates are available in the Response directory on the CD-ROM. To start Oracle Universal Installer and specify the response file, enter the following command at the command line in the directory where the Universal Installer is installed:

```
setup.exe -responsefile filename [-silent]
```

Note: This is not character mode.

Non-Interactive Installation Using Response Files

Example: Launch the Oracle Universal Installer in non-interactive mode.

UNIX

```
./runInstaller -responsefile FILENAME [-SILENT] [-NOWELCOME]
```

where:

- **FILENAME:** Identifies the response file
- **SILENT:** Runs Oracle Universal Installer in silent mode
- **NOWELCOME:** Does not display the Welcome window. If using **SILENT**, this parameter is not necessary

Sample Response File for UNIX

```
[General]
```

```
RESPONSEFILE_VERSION=1.7.0
```

```
[Session]
```

```
UNIX_GROUP_NAME="dba"
```

```
FROM_LOCATION="/u01/stage/products.jar"
```

```
ORACLE_HOME="/u01/app/oracle/ora9i"
```

```
ORACLE_HOME_NAME="Ora9i"
```

```
TOPLEVEL_COMPONENT={"oracle.server", "9.0.1.1.1"}
```

```
SHOW_COMPONENT_LOCATIONS_PAGE=false
```

```
SHOW_SUMMARY_PAGE=false
```

```
SHOW_INSTALL_PROGRESS_PAGE=false
```

```
SHOW_REQUIRED_CONFIG_TOOL_PAGE=false
```

```
SHOW_OPTIONAL_CONFIG_TOOL_PAGE=false
```

```
SHOW_END_SESSION_PAGE=false
```

```
NEXT_SESSION=true
```

```
SHOW_SPLASH_SCREEN=true
```

```
SHOW_WELCOME_PAGE=false
```

```
SHOW_ROOTSH_CONFIRMATION=true
```

```
SHOW_EXIT_CONFIRMATION=true
```

```
INSTALL_TYPE="Typical"
```

```
s_GlobalDBName="u01.us.oracle.com"
```

```
s_mountPoint="/u01/app/oracle/ora9i/dbs"
```

```
s_dbSid="db09"
```

```
b_createDB=true
```

Non-Interactive Installation Using Response Files (continued)

Sample Response File (continued)

- The General section specifies the version number of the response file.
- The Sessions section lists various dialogs of the Universal Installer. Some of the dialogs include:
 - FROM_LOCATION: Specifies the location of the source of the products to be installed
 - ORACLE_HOME: Value for ORACLE_HOME
 - ORACLE_HOME_NAME: Value for ORACLE_HOME_NAME
 - SHOW_INSTALL_PROGRESS: The installation progress page that appears during the installation phase
 - SHOW_ROOTISH_CONFIRMATION: Set to TRUE if the confirmation dialog to run the `root.sh` script needs to be shown
 - SHOW_EXIT_CONFIRMATION: Set to TRUE if the confirmation when exiting the installer needs to be shown

The success or failure of a silent installation is generated in a file called `silentInstall.log`.

UNIX

This file will be generated in the `/tmp` directory.

NT

This file will be generated in the directory specified by the `TEMP` variable.

Note: Refer to the operating system–specific installation guide for complete details about setting up a response file.

Oracle Database Configuration Assistant

You use the Oracle Database Configuration Assistant to:

- **Create a database**
- **Configure database options**
- **Delete a database**
- **Manage templates**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Oracle Database Configuration Assistant

Creating a database using Oracle Database Configuration Assistant is covered in the “Creating a Database” lesson.

Database Administrator Users

- **Users `SYS` and `SYSTEM` are created automatically**
 - During database creation
 - Granted the DBA role
- **User `SYS`**
 - Owner of the database data dictionary
 - Default password: `change_on_install`
- **User `SYSTEM`**
 - Owner of additional internal tables and views used by Oracle tools
 - Default password: `manager`

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Database Administrator Users

Extra privileges are necessary to execute administrative duties on the Oracle server, such as creating users. Two database user accounts, `SYS` and `SYSTEM`, are created automatically with the database and granted the DBA role. That is, a predefined role that is created automatically with every database. The DBA role has all database system privileges.

`SYS`

When a database is created, the user `SYS`, identified initially by the password `change_on_install`, is created. `SYS` owns the vitally important data dictionary. When connecting as `SYS` it should be made as `SYSDBA` or `SYSOPER`. If you attempt to connect without `SYSDBA` or `SYSOPER` privileges, you will receive the error: ORA-28009 connecting to `SYS` should be `SYSDBA` or `SYSOPER`.

Database Administrator Users (continued)

SYSTEM

When a database is created, the user `SYSTEM` is also automatically created. `SYSTEM` is identified initially by the password, `manager`. Additional tables and views owned by the user `SYSTEM` are also created. They contain administrative information used by Oracle tools.

Additional Database Administrator Users

Additional users may be created depending on the mode of database creation, that is, manually or by using Database Configuration Assistant. You should create at least one additional administrator username to use when performing daily administrative tasks.

Default Passwords for `SYS` and `SYSTEM`

For security reasons, the default passwords of `SYS` and `SYSTEM` should be changed immediately after database creation.

Note: Beginning with Oracle9i Release 2, the Database Configuration Assistant will prompt you to specify a password, other than the default, for `SYS` and `SYSTEM`. In addition, when creating a database using the SQL*Plus `CREATE DATABASE` command, the users `SYS` and `SYSTEM` can be identified with passwords other than the default. If passwords are not identified within the `CREATE DATABASE` command, the default passwords will be used.

SQL*Plus

- **An Oracle tool providing:**
 - Capability to interact with and manipulate the database
 - Ability to start up and shut down the database, create and run queries, add rows, modify data, and write customized reports
- **A subset of the standard SQL language with specific add ons**
- **Connecting to SQL*Plus:**

```
sqlplus /nolog  
connect / as sysdba  
Connected to an idle instance.
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

SQL*Plus

SQL*Plus is an Oracle command-line tool used to run the standard SQL (Structured Query Language) suite of commands. SQL is a functional language that you use to communicate with Oracle to retrieve, add, update, or modify data in the database.

Oracle Enterprise Manager

- **Serves as a centralized systems management tool for DBAs**
- **A tool to administer, diagnose, and tune multiple databases**
- **A tool to administer multiple network nodes and services from many locations**
- **Use to share tasks with other administrators**
- **Provides tools for administering parallel servers and replicated databases**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

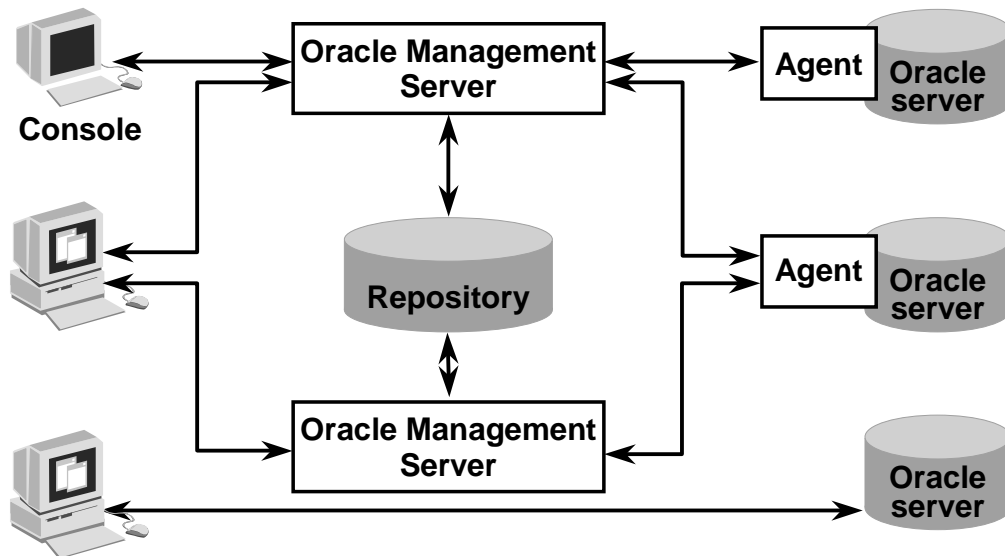
Oracle Enterprise Manager

The Oracle Enterprise Manager is a unified management framework consisting of a Java-based console, a suite of tools and services, and a network of management servers and intelligent agents. It includes both hierarchical tree and graphical representations of the objects and their relationships in the system.

The common services, including job scheduling and management, event management, database discovery and management, service discovery and management, all provide a complete framework for the Oracle Enterprise Manager.

In addition, Oracle Enterprise Manager includes integrated applications that allow you to perform both routine and advanced administration tasks. They include optional packs such as Diagnostics Pack, Tuning Pack, and Change Management Pack, and other applications such as Oracle Net Manager, Spatial Index Advisor, and Text Manager.

Oracle Enterprise Manager: Architecture



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Oracle Enterprise Manager: Architecture

Oracle Enterprise Manager utilizes a three-tier architecture, which includes:

First tier: Console clients and integrated tools provide a graphical interface for administrators.

Second tier: Oracle Management Server and a database repository provide a scalable middle tier for processing system management tasks.

Third tier: Intelligent agents installed on each node monitor its services and execute tasks from the Management Server.

Because not all situations need to implement Oracle Enterprise Manager as a three-tier system, Oracle Enterprise Manager is also available in a two-tier architecture, which connects directly to the databases. The console-launched stand-alone allows a single person to use one or more applications without requiring an Oracle Management Server or intelligent agent.

You use a stand-alone console if you want to perform basic administrative tasks that do not require the job, event, or group system.

Oracle Enterprise Manager: Architecture (continued)

Console

The first tier comprises clients such as consoles and management applications, which present graphical user interfaces to administrators for all management tasks. The first tier depends on the second tier Oracle Management Server for the bulk of its application logic.

Note: As of Oracle9i, connection to the console can be done as stand-alone. Prior to Oracle9i, all connections to the console were made through an Oracle Management Server.

Oracle Management Server

The second tier component of Oracle Enterprise Manager is the Oracle Management Server (OMS). The OMS is the core of the Oracle Enterprise Manager framework and provides administrative user accounts, processes functions such as jobs and events, and manages the flow of information between the console (first tier) and the managed nodes (third tier).

The OMS uses a repository to store all system data, application data, information about the state of the managed nodes, and information about any system-managed packs.

Oracle Enterprise Manager Repository

The repository is a set of tables, created when you set up the OMS. The OMS uses the repository as its persistent back-end store. If necessary, more than one OMS can be used. Multiple OMSs share a repository and provide reliability and fault tolerance.

Nodes

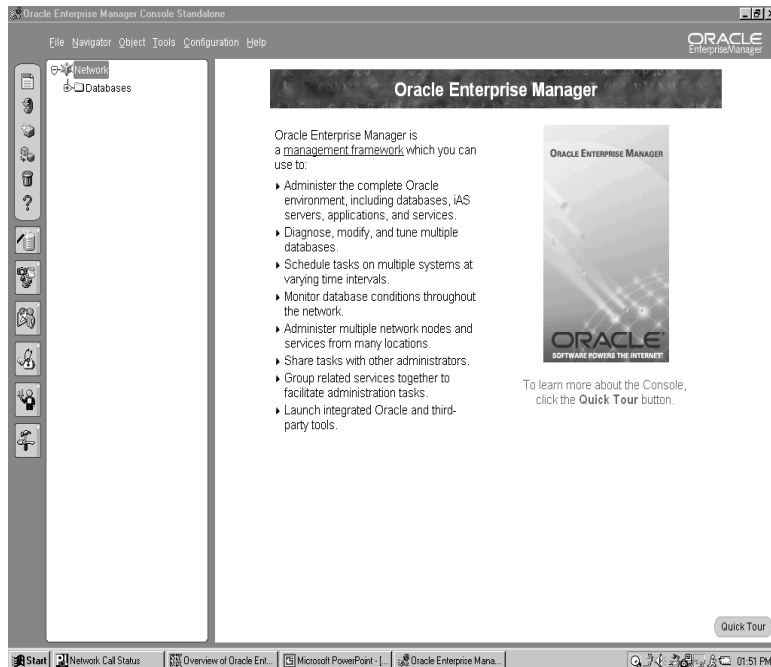
The third tier consists of managed nodes, which contain targets such as databases and other managed services. Residing on each node is an Oracle intelligent agent, which communicates with the OMS and performs tasks sent by consoles and client applications.

Only one intelligent agent is required per node.

An intelligent agent functions independently from the database as well as the console and Oracle Management Server. By running independently of other components, intelligent agents can perform such tasks as starting up and shutting down a database and staying operational if another part of the system is down. The ID for the intelligent agent is `dbstmp`.

Console

- **Central launching point**
- **Can be run in a thin or fat client**
- **Can be launched stand-alone or via an OMS**



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Console

The console provides a graphical interface for administrators and a central launching point for all management applications and tools. In addition, SQL*Plus Worksheet can be launched from the console.

The console can be run either in thin mode through the Web or as a fat client. Thin clients use a Web browser to connect to a server where console files are installed, whereas fat clients require console files to be installed locally.

The console can be launched either in stand-alone mode or by connecting to Oracle Management Server.

Note: This course is not intended to give you details about Oracle Enterprise Manager, the console, or Oracle Management Server. For complete details about using Oracle Enterprise Manager refer to the *Oracle Enterprise Manager 9i* course.

How to Start the Oracle Enterprise Manager Console

Example: Start Oracle Enterprise Manager Console.

1. Launch the console:
Start > Programs > Oracle-OraHome92 > Enterprise Manager Console
2. Start the console by selecting one of the options:
 - Login to the Oracle Management Server
 - Launch standalone
3. Click OK.

Summary

In this lesson, you should have learned to:

- **Identify database administration tools**
- **Identify the features of the Oracle Universal Installer**
- **Use SQL*Plus to interact and manipulate the database**
- **Identify the main components of Oracle Enterprise Manager**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 2 Overview

This practice covers the following topics:

- **Connecting to SQL*Plus**
- **Connecting to Enterprise Manager Console**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 2: Getting Started with the Oracle Server

This practice is instructor led. Your instructor will provide you with login accounts and walk you through logging in to your account. Write the information provided by your instructor below.

Host name: _____

SID name: _____

1 Connect to SQL*Plus as SYSDBA.

Hint

- Connect to your account using instructions provided by your instructor.
- Start SQL*Plus.
- Connect as the user SYS.

```
$ sqlplus /nolog
SQL*Plus: Release 9.2.0.1.0 - Production on Mon Aug 5
10:52:10 2002
Copyright (c) 1982, 2002, Oracle Corporation. All rights
reserved.
SQL> CONNECT / AS SYSDBA
Connected.
```

2 Using SQL*Plus, run the following query to verify that the connection to the database has been made.

```
SQL> SELECT * FROM DUAL;
D
-
X
```

3 Launch Oracle Enterprise Manager in stand-alone mode.

- Navigate to Start > Programs > Oracle-OraHome92 > Enterprise Manager Console.
- Select Launch standalone option.
- Select OK.

Practice 2: Getting Started with the Oracle Server (continued)

- 4 If you are in an Oracle classroom you must perform the following four steps that are particular to the Oracle classroom setup:
 - a. Click the omsconfig file update icon on the desktop.
Enter the name of the UNIX server your class is using. Your instructor will provide the server name. Enter it exactly as it is given to you. This is a case-sensitive entry.
 - b. Open an MSDOS window.
 - c. At the command prompt enter: `oemctl start oms`. Wait for the message:
"The Oracle92_homeManagementServer service was started successfully."
 - d. Close the MSDOS window.

Launch Oracle Enterprise Manager using Oracle Management Server.

- Start the OEM Console and select the Login to the Oracle Management Server option. Log in as follows:
Administrator: `sysman` **Note:** Case is important.
Password: `oem_temp` **Note:** Case is important.
When prompted, change the password to `oracle`. **Note:** Case is important.
Management Server: (Instructor supplied)
- Navigate to Navigator > Discover Nodes from the main menu after the OEM. Console is opened. The Discovery Wizard dialog box will appear.
- Select Next.
- Enter the name of the node you want to manage: that is, designated database server host name. (Instructor supplied)
- Click Next.
- Select Finish after discovery is complete.
- Click OK. **Note:** Alert the instructor if discovery is not successful.
- Expand the Database folder.
- Double click your designated database provided by your instructor.
- Enter the connection information:
User: `sys`
Password: `secure`
As: `SYSDBA`
- Set the node credentials for running jobs.
- Navigate to Configuration > Preferences from the main menu
- Select the Preferred Credentials page.
- Scroll down and select the entry for your designated database.
- Supply the following:
Username: (Instructor supplied)
Password: (Instructor supplied)
Confirm Password.
Role: `SYSDBA`
- Click OK.

Practice 2: Getting Started with the Oracle Server (continued)

5 Start SQL*Plus Worksheet.

SQL*Plus Worksheet can be started within the Oracle Enterprise Manager Console using the following navigation:

Navigate to Tools > Database Applications > SQL*Plus Worksheet

SQL*Plus Worksheet can also be started from the Windows NT menu by doing the following:

Navigate to Start > Programs > Oracle-OraHome92 > Application Development > SQLPlus Worksheet

- Connect directly to the database defined by the instructor, enter:
 - Username
 - Password
 - Service
- Connect as: SYSDBA
- Click OK.

Note: Each time you log in as a different user (within SQL*Plus Worksheet), the service name must be included in the connection string.

3

Managing an Oracle Instance

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

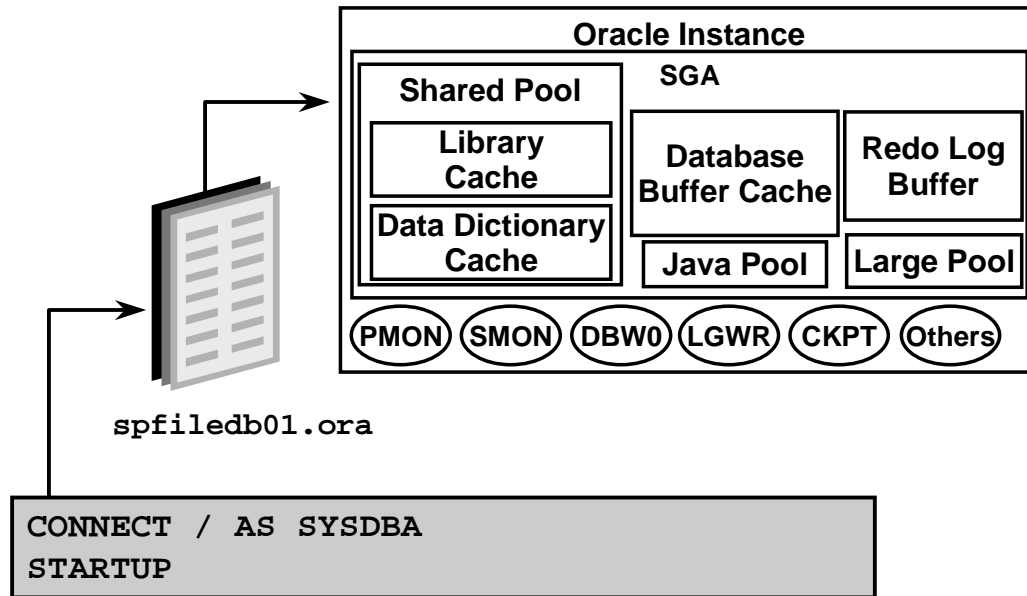
After completing this lesson, you should be able to do the following:

- **Create and manage initialization parameter files**
- **Start up and shut down an instance**
- **Monitor and use diagnostic files**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Initialization Parameter Files



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Initialization Parameter Files

In order to start an instance and open the database, you must connect as SYSDBA and enter the `STARTUP` command. The Oracle server will then read the initialization parameter file and prepare the instance according to the initialization parameters contained within.

Note: You must have SYSDBA privilege. Authentication and the SYSDBA privilege will be discussed in later lessons.

Initialization Parameter Files

- **Entries are specific to the instance being started**
- **Two types of parameters:**
 - **Explicit: Having an entry in the file**
 - **Implicit: No entry within the file, but assuming the Oracle default values**
- **Multiple initialization parameter files can exist**
- **Changes to entries in the file take effect based on the type of initialization parameter file used:**
 - **Static parameter file, PFILE**
 - **Persistent server parameter file, SPFILE**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Initialization Parameter Files

To start an instance, the Oracle server reads the initialization parameter file. Two types of initialization parameter files exist:

- Static parameter file, PFILE, commonly referred to as `initSID.ora`.
- Persistent server parameter file, SPFILE, commonly referred to as `spfileSID.ora`.

Initialization Parameter File Contents

- A list of instance parameters
- The name of the database the instance is associated with
- Allocations for memory structures of the System Global Area (SGA)
- What to do with filled online redo log files
- The names and locations of control files
- Information about undo segments

Multiple initialization parameter files can exist for an instance in order to optimize performance in different situations.

Initialization Parameter Files (continued)

Using Oracle Enterprise Manager to View Initialization Parameters

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Select All Initialization Parameters from the General tabbed page.

PFILE

initSID.ora

- **Text file**
- **Modified with an operating system editor**
- **Modifications made manually**
- **Changes take effect on the next start up**
- **Only opened during instance start up**
- **Default location is \$ORACLE_HOME/dbs**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

PFILE

The PFILE is a text file that can be maintained using a standard operating system editor. The PFILE is read only during instance startup. If the file is modified, the instance must be shut down and restarted in order to make the new parameter values effective.

By default, the PFILE is located in the \$ORACLE_HOME/dbs directory and named initSID.ora.

Creating a PFILE

- **Created from a sample `init.ora` file**
 - Sample installed by the Oracle Universal Installer
 - Copy sample using operating system copy command
 - Uniquely identified by database SID

```
cp init.ora $ORACLE_HOME/dbs/initdba01.ora
```

- **Modify the `initSID.ora`**
 - Edit the parameters
 - Specific to database needs

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a PFILE

A sample `init.ora` file is created by the Universal Installer during installation. This sample `init.ora` file can be used to create an instance-specific `initSID.ora`. A text editor can be used to modify the parameters within the `initSID.ora` file.

PFILE Example

```
# Initialization Parameter File: initdba01.ora
db_name           = dba01
instance_name     = dba01
control_files     = (
    /home/dba01/ORADATA/u01/control01dba01.ctl,
    /home/dba01/ORADATA/u02/control01dba02.ctl)
db_block_size    = 4096
db_cache_size    = 4M
shared_pool_size  = 50000000
java_pool_size   = 50000000
max_dump_file_size = 10240
background_dump_dest = /home/dba01/ADMIN/BDUMP
user_dump_dest   = /home/dba01/ADMIN/UDUMP
core_dump_dest   = /home/dba01/ADMIN/CDUMP
undo_management   = AUTO
undo_tablespace  = UNDOTBS
. . .
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

PFILE Example

- Specify the values in the following format: keyword=value.
- The server has a default value for each parameter. This value may be operating system dependent, depending on the parameter.
- Parameters can be specified in any order, although there are some exceptions.
- Comment lines begin with the # symbol.
- Enclose parameters in double quotation marks to include character literals.
- Additional files can be included with the keyword IFILE.
- If case is significant for the operating system, then it is also significant in filenames.
- Multiple values are enclosed in parentheses and separated by commas.

Note: Develop a standard for listing parameters; either list them alphabetically or group them by functionality. The PFILE varies from instance to instance and does not necessarily look like the preceding example.

SPFILE

spfileSID.ora

- **Binary file**
- **Maintained by the Oracle server**
- **Always resides on the server side**
- **Ability to make changes persistent across shut down and start up**
- **Can self-tune parameter values**
- **Can have Recovery Manager support backing up to the initialization parameter file**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

SPFILE

An SPFILE, new to Oracle9i, is a binary file. The file is not meant to be modified manually and must always reside on the server side. After the file is created it is maintained by the Oracle server. If modified manually, the SPFILE is rendered useless. The SPFILE provides the ability to make changes to the database persistent across shutdown and startup. It also provides the ability to self-tune parameter values, which are recorded in the file. RMAN support for backing up the initialization parameter file is possible because the SPFILE resides on the server side. By default, the file is located in \$ORACLE_HOME/dbs and has a default name in the format of spfileSID.ora.

Creating an SPFILE

- Created from a PFILE file

```
CREATE SPFILE = '$ORACLE_HOME/dbs/spfileDBA01.ora'  
FROM PFILE = '$ORACLE_HOME/dbs/initDBA01.ora';
```

where

- SPFILE-NAME: SPFILE to be created
- PFILE-NAME: PFILE creating the SPFILE

- Can be executed before or after instance start up

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating an SPFILE

An SPFILE is created from a PFILE file using the CREATE SPFILE command. This command requires the SYSDBA privilege to execute. This command can be executed before or after instance startup.

```
CREATE SPFILE [= 'SPFILE-NAME' ]  
FROM PFILE [= 'PFILE-NAME' ] ;
```

where:

- SPFILE-NAME: Name of the SPFILE to be created
- PFILE-NAME: Name of the PFILE being used to create the SPFILE. The PFILE must be available on the server side

If the SPFILE-NAME and PFILE-NAME are not included in the syntax, Oracle will use the default PFILE to generate an SPFILE with a system-generated name.

```
SQL> CREATE SPFILE FROM PFILE;
```


Creating an SPFILE (continued)

Exporting an SPFILE

The contents of an SPFILE can be exported into a PFILE.

```
SQL> CREATE PFILE FROM SPFILE;
```

The PFILE is created as a text file on the server side. This command can be executed either before or after instance startup. This provides an easy way to view the SPFILE and make modifications by:

- Exporting the SPFILE to a PFILE
- Editing the PFILE
- Re-creating the SPFILE from the edited PFILE

Exporting an SPFILE to a PFILE can also serve as another alternative to creating a backup of the server parameter file.

Note: With Oracle9i, RMAN can also back up server parameter files.

V\$SPPARAMETER

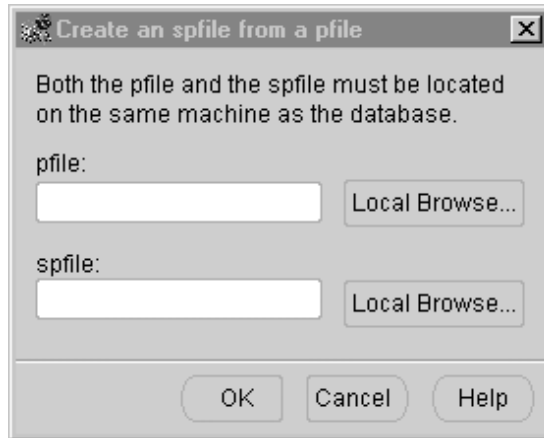
As shown above, there are several options for viewing the parameter settings within the SPFILE. V\$SPPARAMETER is another source for presenting and viewing contents of the SPFILE.

Creating an SPFILE

Using Oracle Enterprise Manager to Create an SPFILE

From the OEM Console:

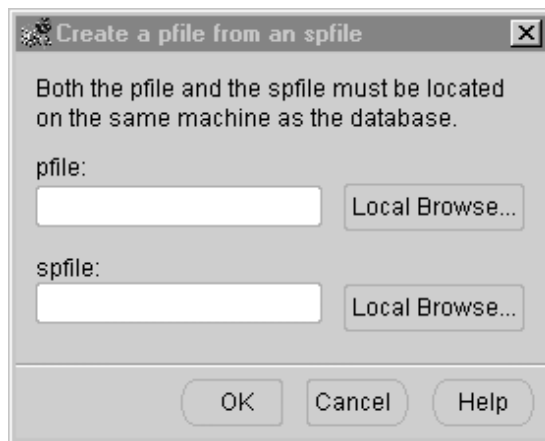
1. Navigate to Instance > Configuration
2. Highlight Configuration.
3. Select Object > Create spfile from the main menu.



Using Oracle Enterprise Manager to Export an SPFILE

From the OEM Console:

1. Navigate to Instance > Configuration
2. Highlight Configuration.
3. Select Object > Create pfile from the main menu.



SPFILE Example

```
*.background_dump_dest= '/home/dba01/ADMIN/BDUMP'  
*.compatible='9.2.0'  
*.control_files='/home/dba01/ORADATA/u01/ctrl01.ctl'  
*.core_dump_dest= '/home/dba01/ADMIN/CDUMP'  
*.db_block_size=4096  
*.db_name='dba01'  
*.db_domain= 'world'  
*.global_names=TRUE  
*.instance_name='dba01'  
*.remote_login_passwordfile='exclusive'  
*.java_pool_size=50000000  
*.shared_pool_size=50000000  
*.undo_management='AUTO'  
*.undo_tablespace='UNDOTBS'  
. . .
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

SPFILE Example

The comments specified on the same lines as a parameter setting in the PFILE are maintained in the SPFILE. All other comments are ignored.

Although the text of an SPFILE is easily viewed in UNIX, the SPFILE is binary, and manual modification of the SPFILE will render it unusable. If you need to view the specific contents of an SPFILE or make some modification, export the SPFILE to a PFILE.

Modifying Parameters in SPFILE

- Changing parameter values

```
ALTER SYSTEM SET undo_tablespace = UNDO2;
```

- Specifying temporary or persistent changes

```
ALTER SYSTEM SET undo_tablespace = UNDO2  
SCOPE=BOTH;
```

- Deleting or resetting values

```
ALTER SYSTEM RESET undo_suppress_errors  
SCOPE=BOTH SID='*';
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Modifying Parameters in SPFILE

The ALTER SYSTEM SET command is used to change the value of instance parameters.

```
ALTER SYSTEM SET parameter_name = parameter_value  
[COMMENT 'text'] [SCOPE = MEMORY|SPFILE|BOTH]  
[SID= 'sid'|'*']
```

where

parameter_name: Name of the parameter to be changed

parameter_value: Value the parameter is being changed to

COMMENT: A comment to be added into the SPFILE next to the parameter being altered

SCOPE: Determines if change should be made in memory, SPFILE, or in both areas

MEMORY: Changes the parameter value only in the currently running instance

SPFILE: Changes the parameter value in the SPFILE only

BOTH: Changes the parameter value in the currently running instance and the SPFILE

SID: Identifies the ORACLE_SID for the SPFILE being used

'sid': Specific SID to be used in altering the SPFILE

'*': Uses the default SPFILE

Modifying Parameters in SPFILE (continued)

Example

```
SQL> SHOW PARAMETERS undo_suppress_errors
NAME                                TYPE                                VALUE
-----
undo_suppress_errors                boolean                             FALSE
```

```
SQL> ALTER SYSTEM SET undo_suppress_errors = TRUE
      2 COMMENT = 'temporary testing' SCOPE=BOTH
      3 SID='DBA01';
```

```
SQL> SHOW PARAMETERS undo_suppress_errors
NAME                                TYPE                                VALUE
-----
undo_suppress_errors                boolean                             TRUE
```

The ALTER SYSTEM RESET command is used to delete or revert to the default value.

```
ALTER SYSTEM RESET parameter_name [SCOPE =
MEMORY|SPFILE|BOTH] [SID= 'sid'|'*']
```

Example

```
SQL> ALTER SYSTEM RESET undo_suppress_errors
      2 SCOPE=BOTH SID='dba01';
```

There are several ways to remove a parameter from the SPFILE:

- Set the parameter back to its default value to simulate deleting using ALTER SYSTEM SET.
- Re-create the SPFILE using CREATE SPFILE FROM PFILE.
- Use ALTER SYSTEM RESET to delete the parameter from the SPFILE.

Modifying Parameters in SPFILE (continued)

Using Oracle Enterprise Manager to Modify the SPFILE Configuration

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Click All Initialization Parameters in the General tabbed page.
4. Modify a parameter in the value column.
5. Click Apply.

STARTUP Command Behavior

- **Order of precedence:**
 - `spfileSID.ora`
 - **Default SPFILE**
 - `initSID.ora`
 - **Default PFILE**
- **Specified PFILE can override precedence.**

```
STARTUP PFILE = $ORACLE_HOME/dbs/initDBA1.ora
```

- **PFILE can indicate to use SPFILE.**

```
SPFILE = /database/startup/spfileDBA1.ora
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

STARTUP Command Behavior

Order of Precedence

- When the command `STARTUP` is used, the `spfileSID.ora` on the server side is used to start up the instance.
- If the `spfileSID.ora` is not found, the default SPFILE on the server side is used to start the instance.
- If the default SPFILE is not found, the `initSID.ora` on the server side will be used to start the instance.

A specified PFILE can override the use of the default SPFILE to start the instance.

A PFILE can optionally contain a definition to indicate use of an SPFILE. This is the only way to start the instance with an SPFILE in a nondefault location.

To start the database with an SPFILE not in the default location:

`SPFILE=<full path and filename>` must be placed in the PFILE.

Example: `SPFILE=$HOME/ADMIN/PFILE/$ORACLE_SID.ora.`

Parameters That Should be Specified in the Initialization Parameter File

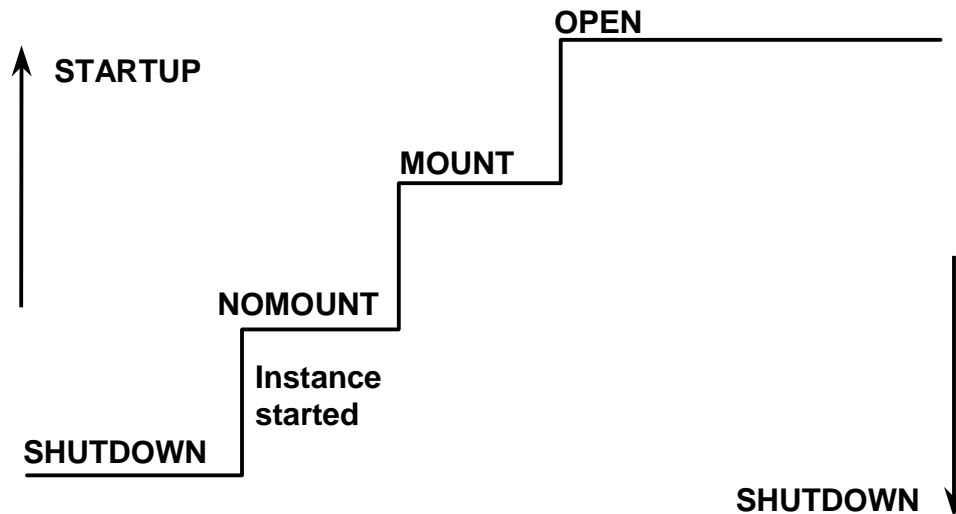
Parameter	Description
BACKGROUND_DUMP_DEST	Location where background process trace files are written (LGWR, DBWn, and so on). Also the location for the alert log file
COMPATIBLE	Version of the server with which this instance should be compatible
CONTROL_FILES	Names of the control files
DB_CACHE_SIZE	Specifies the size of the cache for standard block size buffers
DB_NAME	Database identifier of eight characters or fewer. This is the only parameter that is required when a new database is created
SHARED_POOL_SIZE	Size in bytes of the shared pool
USER_DUMP_DEST	Location where user debugging trace files are created on behalf of a user process

Note: The default values depend on the version of the Oracle server.

Commonly Modified Parameters

Parameter	Description
IFILE	Name of another parameter file to be embedded within the current parameter file. Up to three levels of nesting is possible.
LOG_BUFFER	Number of bytes allocated to the redo log buffer in the SGA
MAX_DUMP_FILE_SIZE	Maximum size of the trace files, specified as number of operating system blocks
PROCESSES	Maximum number of operating system processes that can connect simultaneously to this instance
SQL_TRACE	Enables or disables the SQL trace facility for every user session
TIMED_STATISTICS	Enables or disables timing in trace files and in monitor screens

Starting Up a Database NOMOUNT



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Starting Up a Database

When starting the database, you select the state in which it starts. The following scenarios describe different stages of starting up an instance.

Starting the Instance (NOMOUNT)

An instance would be started in the NOMOUNT stage only during database creation or the re-creation of control files.

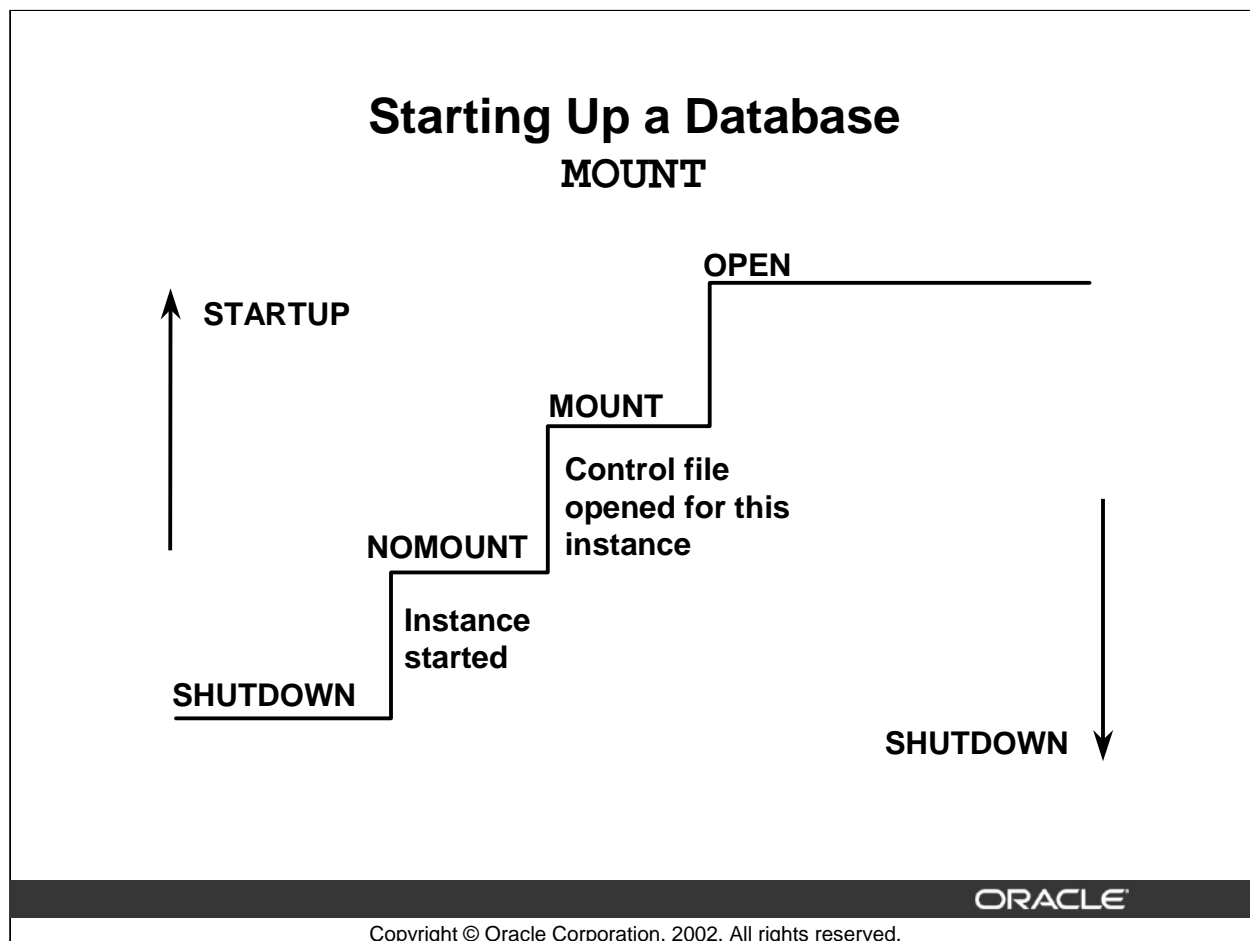
Starting an instance includes the following tasks:

- Reading the initialization file from `$ORACLE_HOME/dbs` in the following order:
 - First `spfileSID.ora`
 - If not found then, `spfile.ora`
 - If not found then, `initSID.ora`

Specifying the `PFILE` parameter with `STARTUP` overrides the default behavior.

- Allocating the SGA
- Starting the background processes
- Opening the `alertSID.log` file and the trace files

The database must be named with the `DB_NAME` parameter either in the initialization parameter file or in the `STARTUP` command.



Starting Up a Database (continued)

Mounting the Database (MOUNT)

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

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

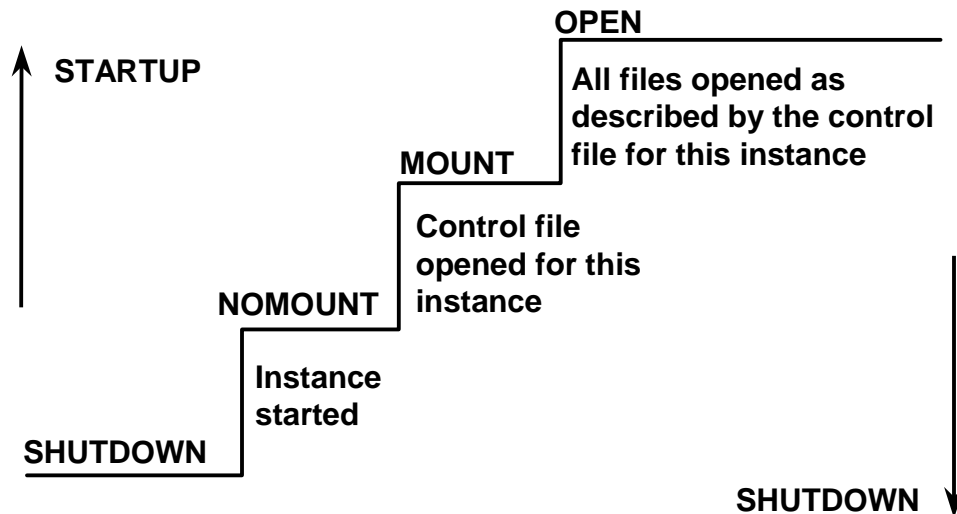
- Renaming data files
- Enabling and disabling online redo log file archiving options
- Performing full database recovery

Mounting a database includes the following tasks:

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

Starting Up a Database

OPEN



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Starting Up a Database (continued)

Opening the Database (OPEN)

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

Opening the database includes the following tasks:

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

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

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

STARTUP Command

Start up the instance and open the database:

```
STARTUP
```

```
STARTUP PFILE=$ORACLE_HOME/dbs/initdb01.ora
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

STARTUP Command

To start up an instance, use the following command:

```
STARTUP [FORCE] [RESTRICT] [PFILE=filename]
        [OPEN [RECOVER][database]
        | MOUNT
        | NOMOUNT]
```

Note: This is not the complete syntax.

where:

- OPEN: Enables users to access the database
- MOUNT: Mounts the database for certain DBA activities but does not provide user access to the database
- NOMOUNT: Creates the SGA and starts up the background processes but does not provide access to the database
- PFILE=*parfile*: Enables a nondefault initialization parameter file to be used to configure the instance

STARTUP Command (continued)

- **FORCE:** Aborts the running instance before performing a normal startup
- **RESTRICT:** Enables only users with **RESTRICTED SESSION** privilege to access the database
- **RECOVER:** Begins media recovery when the database starts

Automating Database Startup

On UNIX:

Automating database start up and shut down can be controlled by the entries in a special operating system file; for example, `oratab` in the `/var/opt/oracle` directory.

Note: Refer to the installation guide of your operating system for more information.

Troubleshooting

If any errors are encountered while issuing the **STARTUP** command the **SHUTDOWN** command must be issued before another **STARTUP**.

Note: **STARTUP** and **SHUTDOWN** commands are **SQL*Plus** commands, not **SQL** commands.

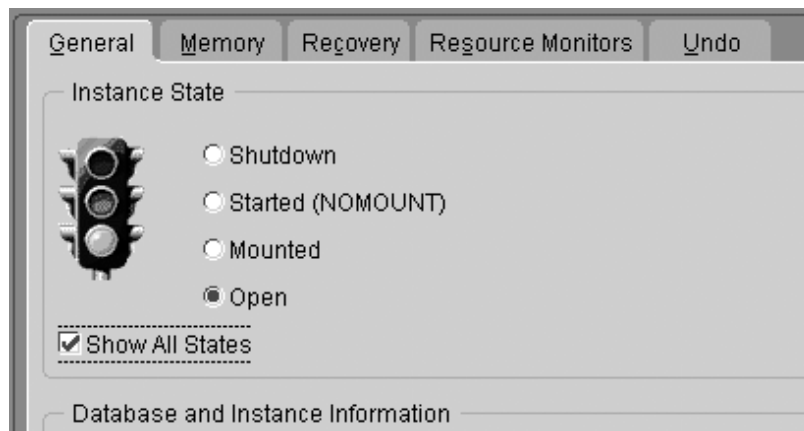
STARTUP Command (continued)

Using Oracle Enterprise Manager to Start Up a Database

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Select the Open option from the General tabbed page.
4. Click Apply.

Note: You must be connected to the database with SYSDBA privileges to perform startup.



ALTER DATABASE Command

- **Change the state of the database from NOMOUNT to MOUNT:**

```
ALTER DATABASE db01 MOUNT;
```

- **Open the database as a read-only database:**

```
ALTER DATABASE db01 OPEN READ ONLY;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

ALTER DATABASE Command

To move the database from the NOMOUNT to a MOUNT stage or from the MOUNT to an OPEN stage, use the ALTER DATABASE command:

```
ALTER DATABASE { MOUNT | OPEN }
```

To prevent data from being modified by user transactions, the database can be opened in read-only mode.

To start up an instance, use the following command:

```
ALTER DATABASE OPEN [READ WRITE | READ ONLY]
```

where:

- **READ WRITE:** Opens the database in read/write mode, so that users can generate online redo log files.
- **READ ONLY:** Restricts users to read-only transactions, preventing them from generating online redo log file information.

Opening a Database in Restricted Mode

- Use the **STARTUP** command to restrict access to a database:

```
STARTUP RESTRICT
```

- Use the **ALTER SYSTEM** command to place an instance in restricted mode:

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Opening a Database in Restricted Mode

A restricted session is useful, for example, when you perform structure maintenance or a database export and import. The database can be started in restricted mode so that it is available only to users with the `RESTRICTED SESSION` privilege.

The database can also be put in restricted mode by using the `ALTER SYSTEM SQL` command:

```
ALTER SYSTEM [ {ENABLE|DISABLE} RESTRICTED SESSION ]
```

where:

- `ENABLE RESTRICTED SESSION`: Enables future logins only for users who have the `RESTRICTED SESSION` privilege
- `DISABLE RESTRICTED SESSION`: Disables `RESTRICTED SESSION` so that users who do not have the privilege can log on

Terminate Sessions

After placing an instance in restricted mode, you may want to kill all current user sessions before performing administrative tasks. This can be done by the following:

```
ALTER SYSTEM KILL SESSION 'integer1,integer2'
```

where:

- `integer1`: Value of the `SID` column in the `V$SESSION` view
- `integer2`: Value of the `SERIAL#` column in the `V$SESSION` view

Opening a Database in Restricted Mode (continued)

Note: The session ID and serial number are used to uniquely identify a session. This guarantees that the `ALTER SYSTEM KILL SESSION` command is applied to the correct session even if the user logs off and a new session uses the same session ID.

Effects of Terminating a Session

The `ALTER SYSTEM KILL SESSION` command causes the background process PMON to perform the following steps upon execution:

- Roll back the user's current transaction.
- Release all currently held table or row locks.
- Free all resources currently reserved by the user.

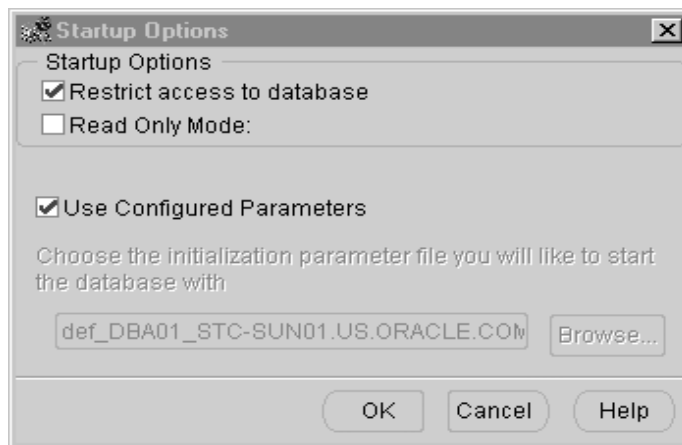
Opening a Database in Restricted Mode (continued)

Using Oracle Enterprise Manager to Open a Database in Restricted Mode

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Select the General tabbed page.
4. Select the Shutdown option under the Instance State.
5. Click Apply. The Shutdown Options dialog box will appear.
6. Select the Immediate option.
7. Click OK.
8. Select Close when processing is complete.
9. Select the Open option under Instance State.
10. Click OK. The Startup Options dialog box will appear.
11. Select the "Restrict access to database" option.
12. Click OK.
13. Click Close when processing complete.

Note: You must be connected to the database with SYSDBA privileges.



Opening a Database in Read-Only Mode

- Opening a database in read-only mode:

```
STARTUP MOUNT  
ALTER DATABASE OPEN READ ONLY;
```

- Can be used to:
 - Execute queries
 - Execute disk sorts using locally managed tablespaces
 - Take data files offline and online, but not tablespaces
 - Perform recovery of offline data files and tablespaces

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Opening a Database in Read-Only Mode

A database can be opened as read-only, as long as it is not already open in read/write mode. The feature is especially useful for a standby database to offload query processing from the production database.

If a query needs to use a temporary tablespace, for example, to do disk sorts, the current user must have a locally managed tablespace assigned as the default temporary tablespace; otherwise, the query fails.

Note: Locally managed tablespaces are discussed in a later lesson.

Read-only mode does not restrict database recovery or operations that change the database state without generating redo data. For example, in read-only mode:

- Data files can be taken offline and online.
- Recovery of offline data files and tablespaces can be performed.

Disk writes to other files, such as control files, operating system audit trails, trace files, and alert log files, can continue in read-only mode.

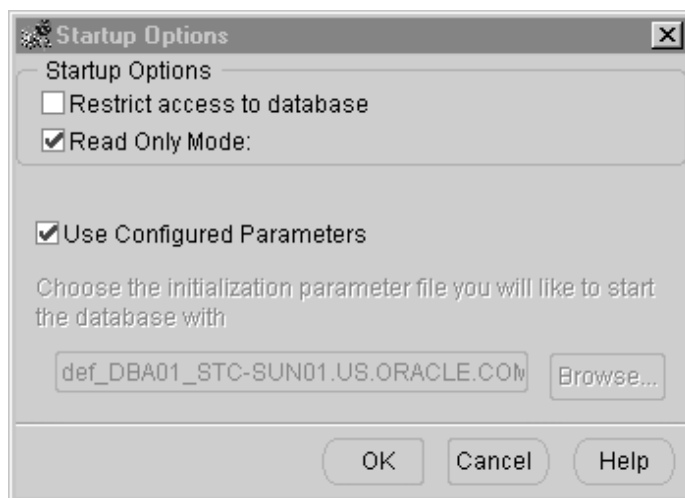
Opening a Database in Read-Only Mode (continued)

Using Oracle Enterprise Manager to Start a Database in Read-Only Mode

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Select the General tabbed page.
4. Select the Shutdown option under Instance State.
5. Click Apply. The Shutdown Options dialog box will appear.
6. Select the Immediate option.
7. Click OK.
8. Select Close when processing is complete.
9. Select the Open option under Instance State.
10. Click OK. The Startup Options dialog box will appear.
11. Select the Read Only Mode check box.
12. Click OK.
13. Click Close when processing complete.

Note: You must be connected to the database with SYSDBA privileges.



Shutting Down the Database

Shutdown Mode	A	I	T	N
Allow new connections	No	No	No	No
Wait until current sessions end	No	No	No	Yes
Wait until current transactions end	No	No	Yes	Yes
Force a checkpoint and close files	No	Yes	Yes	Yes

Shutdown mode:

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

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

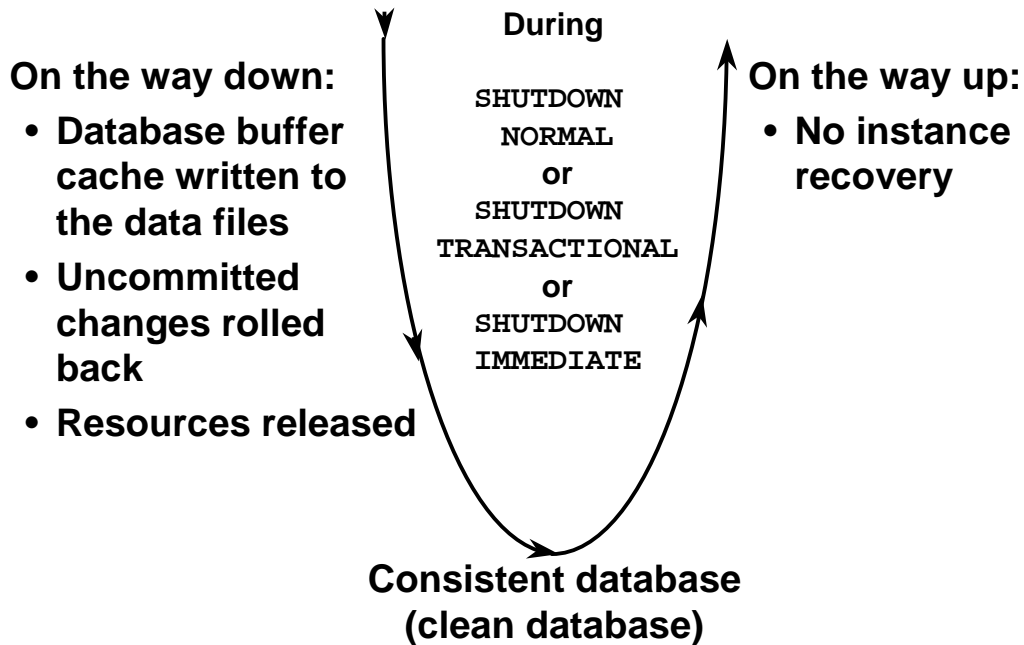
Shutting Down the Database

Shut down the database to make operating system offline backups of all physical structures and to have modified static initialization parameters take effect when restarted.

To shut down an instance you must connect as SYSOPER or SYSDBA and use the following command:

```
SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT ]
```

SHUTDOWN Options



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

SHUTDOWN Options

SHUTDOWN NORMAL

Normal is the default shut down mode. Normal database shut down proceeds with the following conditions:

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

SHUTDOWN TRANSACTIONAL

A transactional shut down prevents clients from losing work. A transactional database shut down proceeds with the following conditions:

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

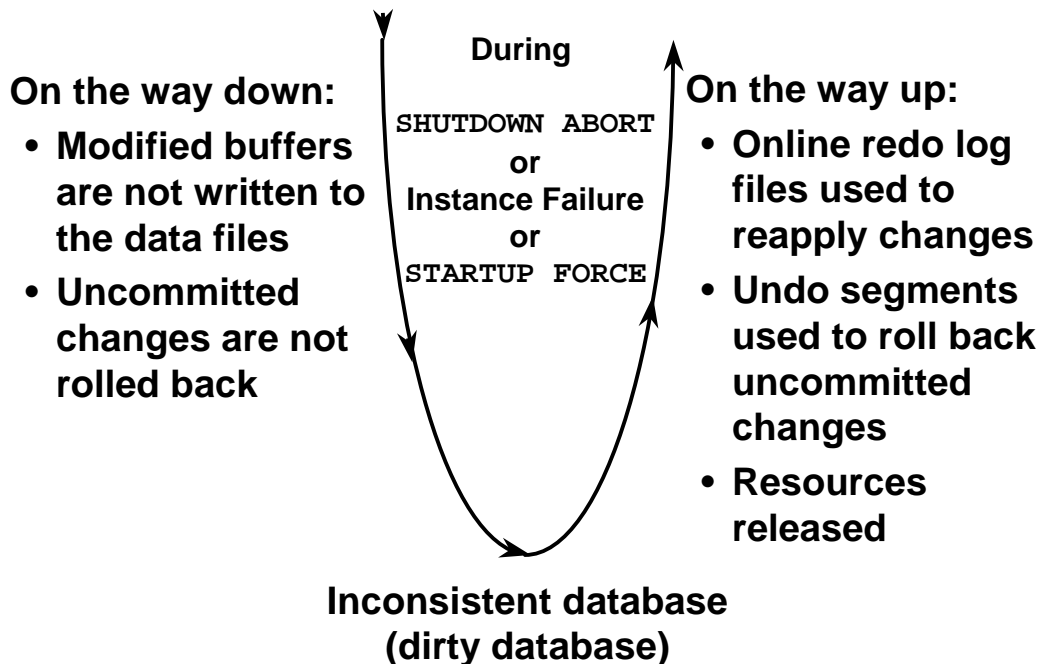
SHUTDOWN Options (continued)

SHUTDOWN IMMEDIATE

Immediate database shut down proceeds with the following conditions:

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

SHUTDOWN Options



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

SHUTDOWN Options (continued)

SHUTDOWN ABORT

If the `NORMAL` and `IMMEDIATE` shut down options do not work, you can abort the current database instance. Aborting an instance proceeds with the following conditions:

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

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

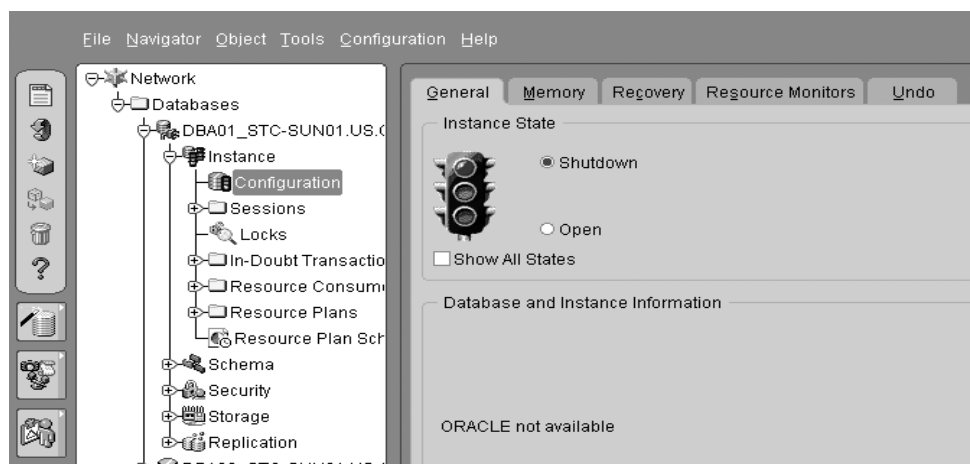
SHUTDOWN Options (continued)

Using Oracle Enterprise Manager to Shut Down a Database

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Select the Shutdown option from the General tabbed page.
4. Click Apply.

Note: You must be connected to the database with SYSDBA privileges to perform a database shut down.



Monitoring an Instance Using Diagnostic Files

- **Diagnostic files**
 - Contain information about significant events encountered
 - Used to resolve problems
 - Used to better manage the database on a day-to-day basis
- **Several types exist:**
 - `alertSID.log` file
 - Background trace files
 - User trace files

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Monitoring an Instance Using Diagnostic Files

Diagnostic files are a means to capture information about the database's activities. They are also useful tools for you when you are managing an instance. Several types exist. The type of diagnostic file created depends on the problem that occurred or the information that is needed to be disseminated.

- Alert log file (`alertSID.log`): Information for day-to-day operation of the database
- Background trace files: Vital information when background processes, such as SMON, PMON, DBWn, and others fail
- User trace files: Vital information for fatal user errors or user forced traced files

Alert Log File

- **alertSID.log file:**
 - Records the commands
 - Records results of major events
 - Used for day-to-day operational information
 - Used for diagnosing database errors
- Each entry has a time stamp associated with it
- Must be managed by DBA
- Location defined by `BACKGROUND_DUMP_DEST`

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Alert Log File

Each Oracle instance has an alert log file. If not already created, it is created during instance start up. The alert log file must be managed by the DBA. It continues to grow while the database continues to work. The alert log file should be the first place you look when diagnosing day-to-day operations or errors. The alert log file also contains pointers to trace files for more detailed information.

The alert log file keeps a record of the following information:

- When the database was started or shut down
- A list of all nondefault initialization parameters
- The start up of background processes
- The thread being used by the instance
- The log sequence number LGWR is writing to
- Information regarding a log switch
- Creation of tablespaces and undo segments
- Alter statements that have been issued
- Information regarding error messages such as ORA-600 and extent errors

Alert Log File (continued)

The `alert_SID.log` location is defined by the `BACKGROUND_DUMP_DEST` initialization parameter.

Background Trace Files

- **Background trace files**
 - Log errors detected by any background process
 - Are used to diagnose and troubleshoot errors
- **Created when a background process encounters an error**
- **Location defined by BACKGROUND_DUMP_DEST**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Background Trace Files

Background trace files are used to log errors that have been encountered by a background process, such as SMON, PMON, DBWn, and other background processes. These files exist only when an error requires writing to the trace files. You use these files to diagnose and troubleshoot problems. Initially when a background trace file is created it contains header information indicating the version number of the data server and the operating system.

Naming convention for user trace file: `sid_processname_PID.trc`
(`db01_lgwr_23845.trc`).

Its location is defined by the `BACKGROUND_DUMP_DEST` initialization parameter.

User Trace Files

- **User trace files**
 - Produced by the user process
 - Can be generated by a server process
 - Contain statistics for traced SQL statements
 - Contain user error messages
- Created when a user encounters user session errors
- Location is defined by `USER_DUMP_DEST`
- Size defined by `MAX_DUMP_FILE_SIZE`

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

User Trace Files

User trace files contain statistics for traced SQL statements, which are useful for SQL tuning. In addition, user trace files contain user error messages.

Naming convention for user trace file:

`sid_orc_PID.trc(db01_orc_23845.trc)`.

Its location is defined by the `USER_DUMP_DEST` initialization parameter.

Enabling or Disabling User Tracing

- **Session level:**
 - Using the `ALTER SESSION` command:
`ALTER SESSION SET SQL_TRACE = TRUE`
 - Executing DBMS procedure:
`dbms_system.SET_SQL_TRACE_IN_SESSION`
- **Instance level**
 - Setting the initialization parameter:
`SQL_TRACE = TRUE`

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Enabling or Disabling User Tracing

Setting `SQL_TRACE=TRUE` at the instance level will produce a significant volume of trace data. This option should be used with care.

User tracing is covered in detail in the *Oracle9i SQL Statement Tuning* course.

Enabling or Disabling User Tracing

Using Oracle Enterprise Manager to Enable or Disable User Tracing

From the OEM Console:

1. Navigate to Instance > Configuration .
2. Highlight Configuration.
3. Select All Initialization Parameters from the General tabbed page.
4. Set the parameter `SQL_TRACE = TRUE`.
5. Click Apply.

Summary

In this lesson, you should have learned how to:

- **Create and manage initialization parameter files**
- **Start up and shut down an instance**
- **Monitor and use diagnostic files**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 3 Overview

This practice covers the following topics:

- **Creating an SPFILE**
- **Starting up and shutting down the database in different modes**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 3 Overview

Note: You can accomplish this practice using SQL*Plus or using OEM and SQL*Plus Worksheet.

Practice 3: Managing an Oracle Instance

- 1 Connect to the database as user SYS and shut down the database.
- 2 With the database shut down, create an SPFILE from the PFILE. The SPFILE will be created in \$ORACLE_HOME/dbs.
- 3 From the operating system, view the SPFILE.
- 4 Connect as user SYS, and start the database using the SPFILE.
- 5
 - a Shut down the database and open it in read-only mode.
 - b Connect as user HR password HR and insert a row into the REGIONS table as follows:

```
INSERT INTO regions VALUES (5, 'Mars');
```


What happens?
 - c Put the database back in read-write mode.
- 6
 - a Connect as user HR password HR and insert the following row into the REGIONS table; do not commit or exit.

```
INSERT INTO regions VALUES (5, 'Mars');
```
 - b In a new telnet session start SQL*Plus. Connect user SYS and perform a SHUTDOWN TRANSACTIONAL.
 - c Roll back the insert in the HR session and exit.
What happens to the HR session?
What happens to the SYS session?
- 7
 - a In the user SYS session start the database.
 - b In the open telnet session start SQL*Plus and connect as user HR.
Note: Keep the two SQL*Plus sessions open, one session as user SYS and one as user HR.
 - c As user SYS enable a restricted session.
 - d As user HR, SELECT from the REGIONS table. Is the SELECT successful?
 - e Exit the session, then reconnect as HR. What happens? The user HR does not have RESTRICTED SESSION privilege, and therefore, cannot log in.
 - f As user SYS disable the restricted session.
 - g Exit the HR telnet session

4

Creating a Database

ORACLE[®]

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **List the prerequisites necessary for database creation**
- **Create a database using Oracle Database Configuration Assistant**
- **Create a database manually**
- **Create a database using Oracle Managed Files**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Planning and Organizing a Database

- **Planning for your database is the first step in managing a database system.**
 - Define the purpose of the database.
 - Define the type of the database.
 - Outline a database architectural design.
 - Choose the database name.
- **Create your database.**
- **Use Oracle Data Migration Assistant to migrate from an earlier version of the database.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Planning and Organizing a Database

Planning for your database is the first step in organizing and implementing a database system. First define how the database will be used. This determines what type of database you need to create that will meet the needs of your business, for example, data warehousing, high online transaction processing, or general purpose. After you determine the purpose and type, you must outline the database architecture that will be used. For example: How will data files, control files, and online redo log files be organized and stored? Oracle's Optimal Flexible Architecture can help you to organize your database file structure and locations. After defining your architecture, you must choose a database and a system identification name for your new database.

Creating your database is a task that prepares several operating system files and is needed only once no matter how many data files the database will use.

During migration from an older version of Oracle, database creation is necessary only if an entirely new database is needed. Otherwise you can use a migration utility. The Oracle Data Migration Assistant is a tool designed to assist you in migrating your current database system.

Optimal Flexible Architecture (OFA)

- **Oracle's recommended standard database architecture layout**
- **OFA involves three major rules:**
 - **Establish a directory structure where any database file can be stored on any disk resource.**
 - **Separate objects with different behavior into different tablespaces.**
 - **Maximize database reliability and performance by separating database components across different disk resources.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Optimal Flexible Architecture (OFA)

Installation and configuration on all supported platforms complies with Optimal Flexible Architecture (OFA). OFA organizes database files by type and usage. Binary files, control files, online redo log files, and administrative files can be spread across multiple disks.

A consistent naming convention provides the following benefits:

- Database files can be easily differentiated from other files.
- It is easy to identify control files, online redo log files, and data files.
- Administration of multiple Oracle homes on the same machine by separating files on different disks and directories becomes easier.
- Better performance is achieved by decreasing disk contention among data files, binary files, and administrative files which can now reside on separate directories and disks.

Oracle Software and File Locations

Software	Files
<code>oracle_base</code>	<code>oradata/</code>
<code>/product</code>	<code>db01/</code>
<code>/release_number</code>	<code>system01.dbf</code>
<code>/bin</code>	<code>control01.ctl</code>
<code>/dbs</code>	<code>redo0101.log</code>
<code>/rdbms</code>	<code>...</code>
<code>/sqlplus</code>	<code>db02/</code>
<code>/admin</code>	<code>system01.dbf</code>
<code>/inst_name</code>	<code>control01.ctl</code>
<code>/pfile</code>	<code>redo0101.log</code>
	<code>...</code>

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Oracle Software and File Locations

The directory tree above shows an example of an OFA-compliant database.

Optimal Flexible Architecture

Another important issue during installation and creation of a database is organizing the file system so that it is easy to administer growth by adding data into an existing database, adding users, creating new databases, adding hardware, and distributing input/output (I/O) load sufficiently across many drives.

Creation Prerequisites

To create a new database, you must have the following:

- **A privileged account authenticated by one of the following:**
 - Operating system
 - Password file
- **Sufficient memory to start the instance**
- **Sufficient disk space for the planned database**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

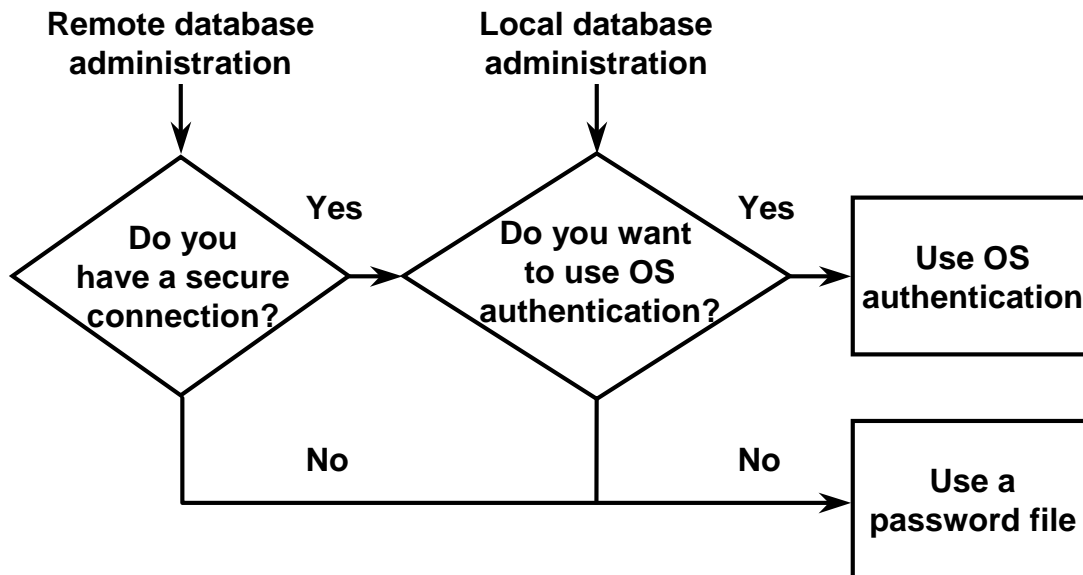
Creation Prerequisites

SYSDBA privileges are required to create a database. These are granted either using operating system authentication or password file authentication.

Before you create the database, make sure that the memory for the SGA, the Oracle executable, and the processes is sufficient. Refer to your operating system installation and administration guides.

Calculate the necessary disk space for the database, including online redo log files, control files, and data files.

Authentication Methods for Database Administrators



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Authentication Methods for Database Administrators

Depending on whether you want to administer your database locally on the same machine on which the database resides or to administer many different database servers from a single remote client, you can choose either operating system or password file authentication to authenticate database administrators.

Note: Refer to operating system–specific manuals for operating system authentication.

Using Password File Authentication

- Create the password file using the password utility.

```
$ orapwd file=$ORACLE_HOME/dbs/orapwU15  
password=admin entries=5
```

- Set `REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE` in initialization parameter file.
- Add users to the password file.
- Assign appropriate privileges to each user.

```
GRANT SYSDBA TO HR;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Using Password File Authentication

Oracle provides a password utility, `orapwd`, to create a password file. When you connect using `SYSDBA` privilege, you are connecting as `SYS` schema and not the schema associated with your username. For `SYSOPER`, you are connected to the `PUBLIC` schema.

Access to the database using the password file is provided by special `GRANT` commands issued by privileged users.

Note: Refer to the “Managing Privileges” lesson for information on grants.

Using Password File Authentication (continued)

Using a Password File

1. Create the password file using the password utility orapwd.

```
orapwd file=filename password=password entries=max_users
```

where:

filename: Name of the password file (mandatory)

password: The password for SYSOPER and SYSDBA (mandatory)

entries: The maximum number of distinct users allowed to connect as SYSDBA or SYSOPER. If you exceed this number, you must create a new password file. It is safer to have a larger number. There are no spaces around the equal-to (=) character.

Example:

```
orapwd file=$ORACLE_HOME/dbs/orapwU15
        password=admin entries=5
```

2. Set the REMOTE_LOGIN_PASSWORDFILE parameter to EXCLUSIVE

where:

EXCLUSIVE: Indicates that only one instance can use the password file and that the password file contains names other than SYS. Using an EXCLUSIVE password file you can grant SYSDBA or SYSOPER privileges to individual users.

3. Connect to the database using the password file created above.

```
CONNECT sys/admin AS SYSDBA
```

Password File Locations

UNIX: \$ORACLE_HOME/dbs

NT: %ORACLE_HOME%/database

Maintaining the Password File

Delete the existing password file using operating system commands, and create a new password file by using the password utility.

Creating a Database

An Oracle database can be created by:

- **Oracle Universal Installer**
- **Oracle Database Configuration Assistant**
 - Graphical user interface
 - Java-based
 - Launched by the Oracle Universal Installer
 - Can be used as a stand-alone application
- **The CREATE DATABASE command**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Database

Creating a database can be performed in one of three ways: can be automatically created as part of the Oracle9i installation using the Oracle Universal Installer, by using the Oracle Database Configuration Assistant (DBCA), or by creating a SQL script using the CREATE DATABASE command.

The Database Configuration Assistant is a graphical user interface that interacts with the Oracle Universal Installer, or can be used stand-alone, to simplify the creation of a database. The DBCA is Java-based and can be launched from any platform with a Java engine.

During the installation of the Oracle Server, DBCA is launched by the Oracle Universal Installer and can automatically create a starter database for you. You have the option of using or not using DBCA, and you also have the option to create a starter database. You also have the option to launch DBCA later as a stand-alone application to create a database.

You can also migrate or upgrade an existing database if you are using a previous version of Oracle software.

Operating System Environment

Set the following environment variables:

- **ORACLE_BASE**
- **ORACLE_HOME**
- **ORACLE_SID**
- **ORA_NLS33**
- **PATH**
- **LD_LIBRARY_PATH**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Operating System Environment

Before creating a database manually or with the Database Configuration Assistant, the operating system environment must be properly configured.

ORACLE_BASE: Specifies the directory at the top of the Oracle software. Example:
`/u01/app/oracle`

ORACLE_HOME: Specifies the directory where the Oracle software is installed. The OFA-recommended value is `$ORACLE_BASE/product/release`. Example:
`/u01/app/oracle/product/9.2`

ORACLE_SID: Specifies the instance name and must be unique for Oracle instances running on the same machine

ORA_NLS33: Required when creating a database with a character set other than `US7ASCII`. Example: `$ORACLE_HOME/ocommon/nls/admin/data`

PATH: Specifies the directories that the operating system searches to find executables, such as `SQL*Plus`. The Oracle9i executables are located in `$ORACLE_HOME/bin` and needs to be added to the `PATH` variable.

LD_LIBRARY_PATH: Specifies the directories for the operating system and Oracle library files. Example: `$ORACLE_HOME/lib`

Database Configuration Assistant

With the Database Configuration Assistant you can:

- **Create a database**
- **Configure database options**
- **Delete a database**
- **Manage templates**
 - **Create new templates using predefined template settings**
 - **Create new templates from an existing database**
 - **Delete database templates**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Database Configuration Assistant

Managing templates is new to Oracle9i. Some predefined templates are available for use. You can also use the existing database as a copy to create a new database or template. The database parameters are stored in XML format.

Benefits of using templates:

- Saves time in creating database
- Templates can be shared
- Database options can be changed, if necessary

Refer to Oracle Database Configuration Assistant online help for more information about templates.

Creating a Database Using Database Configuration Assistant

1. Select create database option
2. Specify type of database
3. Specify global database name and SID
4. Select features
5. Select database operational mode
6. Specify options for memory, character sets, database sizing, file locations, and archiving
7. Define database storage parameters
8. Select option to:
 - Create database
 - Save as a template
 - Generate creation script

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Database Using Database Configuration Assistant

Launch the Database Configuration Assistant:

Programs > Oracle-OraHome92 > Configuration and Migration Tools > Database Configuration Assistant.

The tool will take you through the following steps:

1. Operations: Select the Create a database option to create a database.
2. Database Templates: Select the type of database you want to create from the list of predefined templates.
 - Data Warehouse
 - General Purpose
 - Transaction processing
 - New Database
 - OID

Note: Highlight the type of database and use the Show Details option to view what will be created with the database template.

Creating a Database Using Database Configuration Assistant (continued)

The templates can be created with or without data files.

- Includes datafiles? Yes: Contains only the structure of the database. You can specify and change all database parameters.
- Includes datafiles? No: Contains both the structure and the physical data files of the database. All log files and control files are automatically created for the database and you can add or remove control files, online redo log file groups, and change the destination and name of data files. You cannot add or remove data files, tablespaces, or rollback segments. Initialization parameters cannot be changed.

3. Database Identification: Specify a Global Database Name and SID.

4. Database Features: Select the features to use in your database from the Database Features tabbed page, such as:

- Oracle Spatial
- Oracle Ultra Search
- Oracle Label Security
- Oracle Data Mining
- Oracle OLAP
- Example Schemas

The Example Schemas contain scripts for the following types of tables:

- Human Resources
- Order Entry
- Product Media
- Sales History
- Shipping

Click on the Standard database features button to view a list of standard features Oracle always recommends:

- Oracle JVM
- Oracle Intermedia
- Oracle Text
- Oracle XML DB

Identify any scripts to be run after database creation within the Custom Scripts tabbed page.

5. Database Connection Options: Select the mode in which you want the database to operate:

- Dedicated Server mode
- Shared Server mode

Creating a Database Using Database Configuration Assistant (continued)

6. Initialization Parameters: Specify options on each tabbed page.
 - Memory
Choose a Typical or Custom Database
 - Typical: Creates a database with minimal user input. With typical option, you can specify one of the following environments to operate the database: Online Transaction Processing (OLTP), Multipurpose, and Data Warehousing.
 - Typical Custom: Allows you to customize the creation of your database. This option is only for database administrators experienced with advanced database creation procedures.
 - Character Sets
Choose a database character set option
 - Use the default: Based on the language settings of the operating system.
 - Use Unicode (AL32UTF8): Enables you to store multiple language groups.
 - Choose from the list of character sets: Drop down menu of options.
 - DB Sizing
Define the block size and sort area size for the database. Data block size of a database can be specified only at the time of database creation. `SORT_AREA_SIZE` is the maximum amount of memory used for sort operations.
 - File Locations
Define location for the initialization parameter file, if an SPFILE will be used, and a trace file locations. In addition an All Initialization Parameters button is available to make any changes to parameters, and a File Location Variables button is available to make any changes from the defaults.
 - Archive
Specify placing the database in ARCHIVELOG mode and enable online redo log files to be archived before being reused.
7. Database Storage: Specify database storage parameters. This page displays a tree listing and summary view (multi-column lists) to allow you to change and view the following objects: control files, tablespaces, data files, undo segments, and online redo log file groups.
8. Creation Options: Select the option to create the database, save as a template, or generate a script.
 - Create Database: This option creates the database immediately.
 - Save as a Database Template: This option saves the database creation parameters as a template. This template will then be added to the list of available templates.
 - Generate Database Creation Scripts: This option enables you to save the database creation parameters as a script file, for later use.

Creating a Database Manually

- Choose a unique instance and database name
- Choose a database character set
- Set operating system variables
- Create the initialization parameter file
- Start the instance in NOMOUNT stage
- Create and execute the `CREATE DATABASE` command
- Run scripts to generate the data dictionary and accomplish post-creation steps
- Create additional tablespaces as needed

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Database Manually

- Choose a unique instance and database name.
- Choose a database character set.

A database character set must be defined. An optional national character set can also be defined. For example:

- Character set `AL32UTF16`
- National character set `AL16UTF16`

Refer to the “Using Globalization Support” lesson for information about the various sets available for use.

- Set operating system variables.

Four environment variables must be set: `ORACLE_HOME`, `ORACLE_SID`, `PATH`, `LD_LIBRARY_PATH`.

- `ORACLE_HOME`: The top directory in which the Oracle9i server is installed.
- `ORACLE_SID`: A user-definable name assigned to an instance of a database. Used to distinguish different database instances running on one machine
- `PATH`: Defines the directories the operating system search to find executables.
- `LD_LIBRARY_PATH`: Defines the directories in which required library files are stored.

Creating a Database Manually (continued)

- Create the initialization parameter file.
The initialization parameter file is created using the sample `init.ora` file installed during the installation process. Copy the sample `init.ora` and name it `initSID.ora`. Make modifications to the file specific to the needs of the database you will be creating. If an SPFILE is to be used, the PFILE must be created first. Refer to the “Managing an Oracle Instance” lesson for instructions on how to create a database specific `initSID.ora` file and an SPFILE.
- Start the instance in NOMOUNT.
Connect as user SYS with SYSDBA privilege. The database must be placed in the NOMOUNT state in order to create a database. Refer to the “Managing an Oracle Instance” lesson for directions on how to place the database in a NOMOUNT state.
- Create and execute the CREATE DATABASE command.
 - Create an SQL script that contains the CREATE DATABASE command. Connect to SQL*Plus as the SYS user with the SYSDBA privilege. With the database in NOMOUNT state, execute the script.
 - The CREATE DATABASE command will be dramatically simplified if the database being created is to use Oracle Managed Files (OMF) to manage the operating system files. Refer to the “Managing an Oracle Instance” lesson for information regarding OMF.
- Run scripts.
 - Two scripts `catalog.sql` and `catproc.sql` must be run after the database is created. Both scripts must be run as the user SYS with SYSDBA privilege. Before executing the scripts the database must be placed in the OPEN state.
 - `catalog.sql`: Creates the views on the base tables and on the dynamic performance views, and their synonyms. It starts other scripts that create objects for:
 - Basic PL/SQL environment, including declarations for PL/SQL data types, predefined exceptions, built-in procedures and functions, SQL operations
 - Auditing
 - Import/Export
 - SQL*Loader
 - Installed options

Creating a Database Manually (continued)

- Run scripts (continued)
 - `catproc.sql`: Creates the packages and procedures required to use PL/SQL. In addition, it creates several of the PL/SQL packages that are used to extend RDBMS functionality. This script also creates additional packages views for alerts, pipes, logminer, large objects, objects, queuing, replication, and other built-in options.
 - `pupbld.sql`: Creates the `Product User Profile` table and related procedures. Running this script will prevent a warning message each time a user connects to SQL*Plus.

Note: This script must be run as user `SYSTEM`.

- Create additional tablespaces. You should create any additional tablespaces required to meet your database needs.

Note: “Appendix A” provides step-by-step instructions for creating a database manually in a UNIX environment. In addition, refer to your operating system–specific Oracle documentation for information about creating databases on your specific platform.

Creating a Database Using Oracle Managed Files (OMF)

- Using OMF simplifies file administration on the operating system.
- OMF are created and deleted by the Oracle server as directed by SQL commands.
- OMF are established by setting two parameters:
 - `DB_CREATE_FILE_DEST`: Set to give the default location for data files
 - `DB_CREATE_ONLINE_LOG_DEST_n`: Set to give the default locations for online redo log files and control files
- Maximum of five locations

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Database Using Oracle Managed Files (OMF)

OMF simplify file administration by eliminating the need to directly manage the files within an Oracle database.

OMF are named as follows:

- Control files: `ora_%u.ctl`
- Online redo log files: `ora_%g_%u.log`
- Data files: `ora_%t_%u.dbf`
- Temporary data files: `ora_%t_%u.tmp`

The following characters are defined as:

- `%u`: An eight-character string that guarantees uniqueness.
- `%t`: The tablespace name, truncated if necessary to fit into the maximum length file name. Placing the tablespace name before the uniqueness string means that all the data files for a tablespace appear next to each other in an alphabetic file listing.
- `%g`: The online redo log file group number.
- `ora_` with `.dbf` at its end: Identifies the file as an OMF.

Creating a Database Using Oracle Managed Files (OMF) (continued)

Undo files do not have a special extension.

Both of the `DB_CREATE_FILE_DEST` and `DB_CREATE_ONLINE_LOG_DEST_n` parameters do not have to be set. One or the other or both can be used.

Creating a Database Using Oracle Managed Files (OMF)

- Define the OMF parameters in the initialization parameter file. Example:

- DB_CREATE_FILE_DEST=/\$HOME/ORADATA/u05
 - DB_CREATE_ONLINE_LOG_DEST_1=/\$HOME/ORADATA/u01
 - DB_CREATE_ONLINE_LOG_DEST_2=/\$HOME/ORADATA/u02

- CREATE DATABASE command is simplified:

```
@cddba01.sql  
> CREATE DATABASE dba01;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Database Using Oracle Managed Files (OMF)

To create a database using OMF the DB_CREATE_FILE_DEST and DB_CREATE_ONLINE_LOG_DEST_n parameters are defined in the initialization parameter file. Once the OMF parameters are defined, the syntax for creating the database becomes simplified since no file names or locations need to be defined.

The example above creates a database with the following OMFs:

- A SYSTEM tablespace datafile in directory /\$HOME/ORADATA/u05 that is 100 MB and autoextensible up to an unlimited size.
- Two online redo log file groups with two members of 100 MB each, one each in /\$HOME/ORADATA/u01 and /\$HOME/ORADATA/u02.
- If automatic undo management mode is enabled, then an undo tablespace data file in directory /\$HOME/ORADATA/u05 that is 10 MB and autoextensible up to an unlimited size. An undo tablespace named SYS_UNDOTBS is created.
- If no CONTROL_FILES initialization parameter was specified, then two control files are created, one each in /\$HOME/ORADATA/u01 and /\$HOME/ORADATA/u02. The control file in /\$HOME/ORADATA/u01 is the primary control file.

Creating a Database Using Oracle Managed Files (OMF) (continued)

If the `CREATE DATABASE` command fails, any OMF created are removed. The internally generated filenames can be seen when the user selects from `DBA_DATAFILES`, `V$DATAFILE`, `V$CONTROLFILE`, and `V$LOGFILE`.

Both `DB_CREATE_FILE_DEST` and `DB_CREATE_ONLINE_LOG_DEST_#` can be modified dynamically with the `ALTER SYSTEM SET` command.

CREATE DATABASE Command

```
CREATE DATABASE user01
  USER SYS IDENTIFIED BY ORACLE
  USER SYSTEM IDENTIFIED BY MANAGER
  CONTROLFILE REUSE
  LOGFILE
    GROUP 1 ('$HOME/ORADATA/u01/redo01.log') SIZE 100M,
    GROUP 2 ('$HOME/ORADATA/u02/redo02.log') SIZE 100M,
    GROUP 3 ('$HOME/ORADATA/u03/redo03.log') SIZE 100M
  MAXLOGFILES 5
  MAXLOGMEMBERS 5
  MAXLOGHISTORY 1
  MAXDATAFILES 100
  MAXINSTANCES 1
  ARCHIVELOG
  FORCE LOGGING
  CHARACTER SET US7ASCII
  NATIONAL CHARACTER SET AL16UTF16
  DATAFILE '$HOME/ORADATA/u01/system01.dbf' SIZE 325M
  DEFAULT TEMPORARY TABLESPACE temp
  UNDO TABLESPACE undotbs
  SET TIME_ZONE= 'America/New_York'
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

CREATE DATABASE Command

To create a database, you must have the SYSDBA system privilege. Use the following SQL command to create the database manually.

Note: Not all of the clauses are not mandatory and have been identified as such. In addition, the example above assumes you have enabled Oracle Managed Files by specifying a value for the DB_CREATE_FILE_DEST parameter in the initialization parameter file. Thus, no file specifications are required for the DEFAULT TEMPORARY TABLESPACE and UNDO TABLESPACE clauses cited above.

CREATE DATABASE Command (continued)

```
CREATE DATABASE database
USER SYS IDENTIFIED BY password
USER SYSTEM IDENTIFIED BY password
CONTROLFILE REUSE
LOGFILE GROUP integer filespec
MAXLOGFILES integer
MAXLOGMEMBERS integer
MAXLOGHISTORY integer
MAXDATAFILES integer
MAXINSTANCES integer
ARCHIVELOG|NOARCHIVELOG
CHARACTER SET charset
NATIONAL CHARACTER SET charset
DATAFILE filespec [autoextend_clause]
    filespec:= 'filename' [SIZE integer][K|M] [REUSE]
    autoextend_clause:= [AUTOEXTEND {OFF|ON [NEXT integer[K|M]]
                        [MAXSIZE {UNLIMITED|integer[K|M]]}]
DEFAULT TEMPORARY TABLESPACE tablespace filespec
    temp_tablespace_extent_clause
    temp_tablespace_extent_clause:=
    EXTENT MANAGEMENT LOCAL UNIFORM [SIZE integer][K|M] ]
UNDO TABLESPACE tablespace DATAFILE filespec [autoextend_clause]
SET TIME_ZONE [time_zone_region
where:
```

- DATABASE: Is the name of the database to be created. The name can be up to 8 bytes long. (If the name of the database is omitted, the initialization parameter DB_NAME is used.)
- USER SYS IDENTIFIED BY and USER SYSTEM IDENTIFIED BY: These clauses were introduced with Oracle9i Database Release 2, and are not mandatory. Use these clauses to establish passwords for the SYS and SYSTEM users. If you specify one clause you must specify the other.
- CONTROLFILE REUSE: Specifies that an existing control file identified in the initialization parameter file should be reused. Typically used when re-creating a database, rather than creating one for the first time.

CREATE DATABASE Command (continued)

- **LOGFILE GROUP:** Specifies the names of the log files to be used and the group to which they belong. The database requires at least two online redo log file groups.
 - If either the `DB_CREATE_ONLINE_LOG_DEST_n` or `DB_CREATE_FILE_DEST` initialization parameters are set (or if both are set), then the Oracle server creates two Oracle Manage Files with system-generated names, 100MB in size, in the default log file directory specified by the these parameters.
 - If neither of the these parameters is set, the Oracle server then creates two online redo log file groups. The names and sizes of the default files will depend on the operating system.
- **MAXLOGFILES:** Specifies the maximum number of online redo log file groups that can ever be created for the database.
- **MAXLOGMEMBERS:** Specifies the maximum number of log file members for a log file group.
- **MAXLOGHISTORY:** Specifies the maximum number of archived redo log files for automatic media recovery of the Oracle Real Application Clusters.
- **MAXDATAFILES:** Specifies the initial sizing of the data file section of the control file at `CREATE DATABASE` or `CREATE CONTROLFILE` time. An attempt to add a new file whose number is greater than `MAXDATAFILES`, but less than or equal to `DB_FILES`, causes the control file to expand.
- **MAXINSTANCES:** Specifies the maximum number of instances that can simultaneously mount and open the database.
- **ARCHIVELOG | NOARCHIVELOG:** Specify `ARCHIVELOG` if you want the contents of an online redo log file group to be archived before the group can be reused. Specify `NOARCHIVELOG` if the contents of an online redo log file group to not need to be archived before the group can be reused.
- **FORCE LOGGING:** Specifies the logging of all changes in the database except for changes in temporary tablespaces and temporary segments.
- **CHARACTER SET:** Specifies the character set the database uses to store data.
- **NATIONAL CHARACTER SET:** Specifies the national character set used to store data in columns defined as `NCHAR`, `NCLOB`, or `NVARCHAR2`. The default is `AL16UTF16`.
- **DATAFILE filespec:** Specifies the data files to be used.
- **DEFAULT TEMPORARY TABLESPACE:** Creates a default temporary tablespace for the database. Oracle will assign to this temporary tablespace any users for whom you do not specify a different temporary tablespace.
- **UNDO TABLESPACE:** Creates an undo tablespace and creates the specified data files as part of the undo tablespace.
- **SET TIME_ZONE:** Specifies the time zone for the database. You can specify the time zone in two ways:
 - Specifying a displacement from UTC (Universal Coordinated Time) Valid range of hh:mm is -12:00 to +14:00
 - Specifying a time zone region. Query `TZNAME` column of the `V$TIMEZONE_NAMES` view to see a listing of valid region names.

Troubleshooting

Creation of the database fails if:

- **There are syntax errors in the SQL script**
- **Files that should be created already exist**
- **Operating system errors such as file or directory permission or insufficient space errors occur**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Troubleshooting

If one of the three problems shown on the slide occurs, the `CREATE DATABASE` statement fails. You should delete any files created by the `CREATE DATABASE` statement, correct the errors, and attempt to create again.

After Database Creation

The database contains:

- Data files, control files, and online redo log files
- User `SYS` with the password `change_on_install`
- User `SYSTEM` with the password manager
- Internal tables (but no data dictionary views)

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

After Database Creation

After the database is created, the instance is left running and the database is open and available for normal use. The database contains users `SYS` and `SYSTEM`. Depending on the method of database creation, either using DBCA or manually, other users might get created. Change the passwords for `SYS` and `SYSTEM` as soon as the database is created.

Note: Beginning with Oracle9i Release 2, DBCA prompts you to enter passwords for `SYS` and `SYSTEM`.

You can view the dynamic performance views such as `V$LOGFILE`, `V$CONTROLFILE`, and `V$DATAFILE`, but no data dictionary views are created.

Summary

In this lesson, you should have learned to:

- **Identify the prerequisites for creating a database**
- **Create a database using the Oracle Database Configuration Assistant**
- **Create a database manually**
- **Create a database using Oracle Managed Files**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 4 Overview

- **This lesson provides two specific ways of creating a database:**
 - **Use the Database Configuration Assistant to create a database using graphical steps. Launched by:
Start > Programs > Oracle-OraHome90 > Configuration and Migration Tools.**
 - **Appendix A provides a step-by- step guide for creating a database manually on a UNIX system.**
- **Review the steps, and optionally create a database manually or by using the Database Configuration Assistant.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

5

Using Data Dictionary and Dynamic Performance Views

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Identify built-in database objects**
- **Identify the contents and uses of the data dictionary**
- **Describe how data dictionary views are created**
- **Identify data dictionary view categories**
- **Query the data dictionary and dynamic performance views**
- **Describe administrative script naming conventions**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Built-In Database Objects

Other objects created with the database:

- Data dictionary
- Performance tables
- PL/SQL packages
- Database event triggers

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

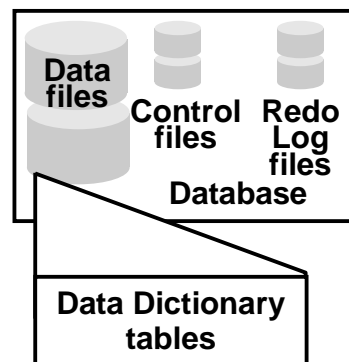
Built-In Database Objects

In addition to creating the database files, several other structures are created.

- Data dictionary: Contains descriptions of the objects in the database
- Dynamic performance tables: Contains information used by the database administrator (DBA) to monitor and tune the database and instance
- PL/SQL packages: Program units adding functionality to the database. These packages are created when the `catproc.sql` script is run after the `CREATE DATABASE` command. PL/SQL packages will not be discussed within the scope of this course.
- Database event triggers: Triggers are procedures that execute implicitly whenever a table or view is modified, or when some user actions or database system actions occur. Database event triggers will not be discussed within the scope of this course.

Data Dictionary

- **Central to every Oracle database**
- **Describes the database and its objects**
- **Contains read-only tables and views**
- **Stored in the `SYSTEM` tablespace**
- **Owned by the user `SYS`**
- **Maintained by the Oracle server**
- **Accessed with `SELECT`**



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Data Dictionary

One of the most important parts of an Oracle database is its data dictionary, which is a read-only set of tables and views that provides information about its associated database.

The data dictionary is updated by the Oracle server whenever a data definition language (DDL) command is executed. In addition, data manipulation language (DML) commands, such as one that causes a table to extend, can update the data dictionary.

Not only is the data dictionary central to every Oracle database, it is an important source of information for all users, from end users to application designers and database administrators.

SQL statements are used to access the data dictionary. Because the database is read-only, you can issue queries only against the tables and views of the data dictionary.

Base Tables and Data Dictionary Views

The data dictionary contains two parts:

- **Base tables**
 - Stores description of the database
 - Created with `CREATE DATABASE`
- **Data dictionary views**
 - Used to simplify the base table information
 - Accessed through public synonyms
 - Created with the `catalog.sql` script

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Base Tables and Data Dictionary Views

The data dictionary contains descriptions of the objects in the database. It includes two types of objects.

Base Tables

Base tables are underlying tables, which store information about the database. The base tables are the first objects created in any Oracle database. They are automatically created when the Oracle server runs the `sql .bsq` script at the time the database is created. Only the Oracle server should write to these tables. Users rarely access them directly, because most of the data is stored in a cryptic format. Never use DML commands to update the base tables directly, with the exception of the `AUD$` table. An example of a base table is the `IND$` table, which contains information about the indexes in the database.

Data Dictionary Views

Data dictionary views are base table summaries, which provide for a more useful display of base table information. For example, in the data dictionary views, object names are used instead of only object numbers. The data dictionary views are created using the `catalog.sql` script which is run after the `CREATE DATABASE` command.

Creating Data Dictionary Views

Script	Purpose
<code>catalog.sql</code>	Creates commonly used data dictionary views and synonyms
<code>catproc.sql</code>	Runs scripts required for server-side PL/SQL

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating Data Dictionary Views

The base tables of the data dictionary are automatically created when a database is created. When the database is created using the Oracle Universal Installer, the scripts to create data dictionary and dynamic performance views, and scripts for your Oracle server options are run automatically.

You must run these scripts manually when creating a new database manually. In addition, you may need to run them again when upgrading to a new release of the Oracle server. These scripts must be run as the user `SYS` with the `SYSDBA` privilege.

The scripts are located in the following directories:

UNIX: `$ORACLE_HOME/rdbms/admin`

NT: `%ORACLE_HOME%\rdbms\admin`

Data Dictionary Contents

The data dictionary provides information about:

- **Logical and physical database structures**
- **Definitions and space allocations of objects**
- **Integrity constraints**
- **Users**
- **Roles**
- **Privileges**
- **Auditing**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Data Dictionary Contents

A data dictionary contains:

- The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)
- How much space has been allocated for, and is currently used by, the schema objects
- Default values for columns
- Integrity constraint information
- The names of Oracle users
- Privileges and roles each user has been granted
- Auditing information, such as who has accessed or updated various schema objects

How the Data Dictionary Is Used

Primary uses:

- **Oracle server uses it to find information about**
 - Users
 - Schema objects
 - Storage structures
- **Oracle server modifies it when a DDL statement is executed.**
- **Users and DBAs use it as a read-only reference for information about the database.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

How the Data Dictionary Is Used

How the Oracle Server Uses the Data Dictionary

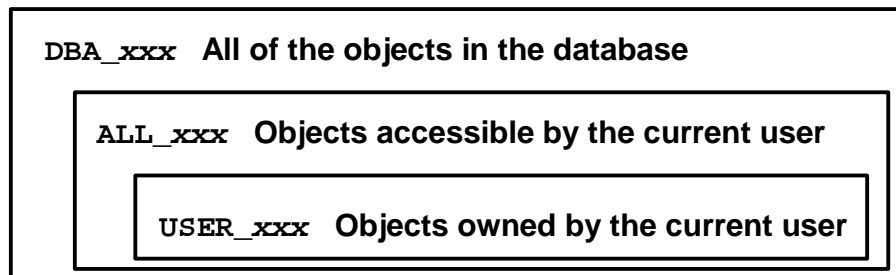
Data in the base tables of the data dictionary are necessary for the Oracle server to function. Therefore, only the Oracle server should write or change data dictionary information. During database operation, the Oracle server reads the data dictionary to ascertain that schema objects exist and that users have proper access to them. The Oracle server also updates the data dictionary continuously to reflect changes in database structures.

How Users and DBAs Use the Data Dictionary

The views of the data dictionary serve as a reference for all database users. Some views are accessible to all Oracle users. Others are intended for database administrators only.

Data Dictionary View Categories

- **Three sets of static views**
- **Distinguished by their scope:**
 - **DBA:** What is in all the schemas
 - **ALL:** What the user can access
 - **USER:** What is in the user's schema



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Data Dictionary View Categories

Views With the DBA Prefix

Views with the DBA prefix show a global view of the entire database. They are meant to be queried only by database administrators. Any user granted the system privilege `SELECT ANY TABLE` can query the DBA prefixed views of the data dictionary.

To query on all objects in the database, the DBA can issue the following statement:

```
SQL> SELECT owner, object_name, object_type
2 FROM dba_objects;
```

Views With the ALL Prefix

Views with the ALL prefix refer to the user's overall perspective of the database. These views return information about schema objects to which the user has access by way of public or explicit grants of privileges and role, in addition to schema objects that the user owns.

For example, the following query returns information about all the objects to which a user has access:

```
SELECT owner, object_name, object_type
FROM all_objects;
```

Data Dictionary View Categories (continued)

Views With the USER Prefix

The views most likely to be of interest to typical database users are those with the USER prefix. These views:

- Refer to the user's own private environment in the database
- Generally refer to objects owned by the current user
- Have columns identical to the other views, except that the column OWNER is implied to be the current user
- Return a subset of the information in the ALL views
- Can have abbreviated public synonyms for convenience

For example, the following query returns all the objects contained in the user's schema:

```
SELECT owner, object_name, object_type
FROM users_objects;
```

Data Dictionary Views

Data dictionary views are static views that answer questions such as:

- Was the object ever created?
- What is the object a part of?
- Who owns the object?
- What privileges do users have?
- What restrictions are on the object?

Note: Refer to the *Oracle9i Database Reference* document for a listing of all data dictionary views.

Data Dictionary Examples

- **General overview:** DICTONARY, DICT_COLUMNS
- **Schema objects:** DBA_TABLES, DBA_INDEXES, DBA_TAB_COLUMNS, DBA_CONSTRAINTS
- **Space allocation:** DBA_SEGMENTS, DBA_EXTENTS
- **Database structure:** DBA_TABLESPACES, DBA_DATA_FILES

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Data Dictionary Examples

To get an overview of the data dictionary views, the DICTONARY view or its synonym DICT can be queried. For example:

```
SQL> SELECT * FROM dictionary;
```

Include the WHERE clause to narrow your responses:

```
SQL> SELECT * FROM dictionary
2 WHERE table_name LIKE 'DBA_SEG%'
```

To get a list of columns within a view, use the DESCRIBE keyword:

```
SQL> DESCRIBE dba_users;
```

To get an overview of the columns in the data dictionary views, you can query the DICT_COLUMNS view.

To view the contents of a data dictionary view, use the SELECT command.

```
SQL> SELECT * FROM dba_users;
```

Note: Refer to the *Oracle9i Database Reference* document for a complete list of data dictionary views and their columns.

Dynamic Performance Tables

- **Virtual tables**
- **Record current database activity**
- **Continually updated while the database is operational**
- **Information is accessed from memory and control file**
- **Used to monitor and tune the database**
- **Owned by SYS user**
- **Synonyms begin with V\$**
- **Listed in V\$FIXED_TABLE**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dynamic Performance Tables

Throughout its operation, the Oracle server records current database activity in a set of virtual tables called dynamic performance views. These virtual tables exist in memory only when the database is running, to reflect real-time conditions of the database operation. They point to actual sources of information in memory and the control file.

These tables are not true tables, and are not to be accessed by most users; however, DBA can query, grant the SELECT privilege, and create views on these views. These views are sometimes called fixed views because they cannot be altered or removed by the DBA.

The dynamic performance tables are owned by SYS, and their names all begin with V_\$. Views are created on these tables, and then public synonyms are created for the views. The synonym names begin with V\$. For example, the V\$DATAFILE view contains information about the database's data files, and the V\$FIXED_TABLE view contains information about all of the dynamic performance tables and views in the database.

The dynamic performance tables answer questions such as:

- Is the object online and available?
- Is the object open?
- What locks are being held?
- Is the session active?

Dynamic Performance Examples

- V\$CONTROLFILE
- V\$DATABASE
- V\$DATAFILE
- V\$INSTANCE
- V\$PARAMETER
- V\$SESSION
- V\$SGA
- \$SPPARAMETER
- V\$TABLESPACE
- V\$THREAD
- V\$VERSION

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dynamic Performance Examples

The V\$FIXED_TABLE view can also be queried to get a listing of the dynamic performance views:

```
SQL> SELECT * FROM V$FIXED_TABLE;
```

To get a list of columns within a view, use the DESCRIBE keyword:

```
SQL> DESCRIBE V$INSTANCE;
```

To view the contents of the view, use the SELECT command.

```
SQL> SELECT * from V$INSTANCE;
```

Dynamic Performance Examples (continued)

Examples

- V\$CONTROLFILE: Lists the names of the control files
- V\$DATABASE: Contains database information from the control file.
- V\$DATAFILE: Contains data file information from the control file
- V\$INSTANCE: Displays the state of the current instance
- V\$PARAMETER: Lists parameters and values currently in effect for the session
- V\$SESSION: Lists session information for each current session
- V\$SGA: Contains summary information on the system global area (SGA)
- V\$SPFILE: Lists the contents of the SPFILE
- V\$TABLESPACE: Displays tablespace information from the control file
- V\$THREAD: Contains thread information from the control file
- V\$VERSION: Version numbers of core library components in the Oracle server

Note: Refer to the *Oracle9i Database Reference* document for a complete list of dynamic performance views and their columns.

Administrative Script Naming Conventions

Convention	Description
<code>cat*.sql</code>	Catalog and data dictionary information
<code>dbms*.sql</code>	Database package specifications
<code>prvt*.plb</code>	Wrapped database package code
<code>utl*.sql</code>	Views and tables for database utilities

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Administrative Script Naming Conventions

The administrative scripts can be separated into categories by their filenames:

`cat*.sql`

These scripts create the data dictionary views. In addition to the `catalog.sql` and `catproc.sql` scripts, there are scripts that create information for Oracle utilities. For example, the `catadtl.sql` script creates data dictionary views for showing metadata information for types and other object features in the ORDBMS. The `catnoadt.sql` script drops these tables and views.

`dbms*.sql` and `prvt*.plb`:

These scripts create objects for predefined Oracle packages that extend the Oracle server functionality. These programs simplify the task of administering the database. Most SQL scripts are run during the execution of the `catproc.sql` script. A few additional scripts must be executed by the database administrator. An example is the `dbmspool.sql` script, which enables you to display the sizes of objects in the shared pool and mark them to be kept or removed in the SGA in order to reduce shared pool fragmentation.

Administrative Script Naming Conventions (continued)

`utl*.sql`

These scripts must be run when the database needs additional views and tables. For example, the `utlxplan.sql` script creates a table used to view the execution plan of a SQL statement.

Note: Most of these scripts must be executed under the user `SYS` with `SYSDBA` privilege. The database administrator should examine the scripts to determine which user account must be used to run the scripts.

Summary

In this lesson, you should have learned how to:

- **Identify built-in database objects**
- **Identify the contents and uses of the data dictionary**
- **Describe how data dictionary views are created**
- **Identify data dictionary view categories**
- **Query the data dictionary and dynamic performance views**
- **Describe administrative script naming conventions**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 5 Overview

This practice covers the following topics:

- **Identifying the components and contents of the data dictionary**
- **Querying the data dictionary and dynamic performance views**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 5 Overview

Note: This practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 5: Using Data Dictionary and Dynamic Performance Views

- 1 Which of the following statements are true about the data dictionary?
 - a The data dictionary describes the database and its objects.
 - b The data dictionary includes two types of objects: base tables and data dictionary views.
 - c The data dictionary is a set of tables.
 - d The data dictionary records and verifies information about its associated database.
- 2 Base tables are created using the `catalog.sql` script.
 - a True
 - b False
- 3 Which three of the following statements are true about how the data dictionary is used?
 - a The Oracle server modifies it when a DML statement is executed.
 - b It is used to find information about users, schema objects, and storage structures.
 - c It is used by users and DBAs as a reference.
 - d The data dictionary is a necessary ingredient for the database to function.
- 4 Data dictionary views are static views.
 - a True
 - b False
- 5 The information for a dynamic performance view is gathered from the control file.
 - a True
 - b False
- 6 Which of the following questions might a dynamic performance view answer?
 - a Is the object online and available?
 - b What locks are being held?
 - c Who owns the object?
 - d What privileges do users have?
 - e Is the session active?

Practice 5: Using Data Dictionary and Dynamic Performance Views

- 7 Connect as SYSTEM/MANAGER and find a list of the data dictionary views.
- 8 Identify the database name, instance name, and size of the database blocks.
Hint: Query the V\$DATABASE, V\$THREAD, and V\$PARAMETER dynamic performance views.
- 9 List the name of the data files.
Hint: Query the V\$DATAFILE dynamic performance view.
- 10 Identify the data file that makes up the SYSTEM tablespace.
Hint: Query the DBA_DATA_FILES data dictionary view to identify the SYSTEM tablespace data file.
- 11 How much free space is available in the database and how much is already used?
Hints
 - Query the DBA_FREE_SPACE data dictionary view to show how much free space is available in the database.
 - Query the DBA_SEGMENTS data dictionary view to display how much space is already used.
- 12 List the name and creation date of the database users.
Hint: Query the DBA_USERS data dictionary view to list the name and the creation of the database users.

6

Maintaining the Control File

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

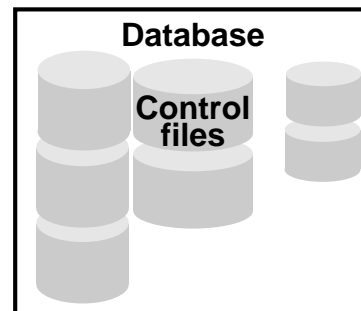
- **Explain the uses of the control file**
- **List the contents of the control file**
- **Multiplex and manage the control file**
- **Manage the control file with Oracle Managed Files (OMF)**
- **Obtain control file information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Control File

- **A small binary file**
- **Defines current state of physical database**
- **Maintains integrity of database**
- **Required:**
 - **At MOUNT state during database start up**
 - **To operate the database**
- **Linked to a single database**
- **Loss may require recovery**
- **Sized initially by
CREATE DATABASE**



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Control File

The control file is a small binary file necessary for the database to start and operate successfully. Each control file is associated with only one Oracle database. Before a database is opened, the control file is read to determine whether the database is in a valid state to use.

A control file is updated continuously by the Oracle server during database use, so it must be available for writing whenever the database is open. The information in the control file can be modified only by the Oracle server; no database administrator or end user can edit the control file.

If for some reason the control file is not accessible, the database does not function properly. If all copies of a database's control files are lost, the database must be recovered before it can be opened.

Control File (continued)

Sizing the Control File

Keywords specified during the creation of the database affect the size of the control file. This is particularly significant when the parameters have large values. The size of the control file is influenced by the following keywords in the `CREATE DATABASE` or `CREATE CONTROLFILE` commands:

- `MAXLOGFILES`
- `MAXLOGMEMBERS`
- `MAXLOGHISTORY`
- `MAXDATAFILES`
- `MAXINSTANCES`

Control File Contents

A control file contains the following entries:

- **Database name and identifier**
- **Time stamp of database creation**
- **Tablespace names**
- **Names and locations of data files and online redo log files**
- **Current online redo log file sequence number**
- **Checkpoint information**
- **Begin and end of undo segments**
- **Redo log archive information**
- **Backup information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Control File Contents

The information in the control file includes the following:

- Database name is taken from either the name specified by the initialization parameter `DB_NAME` or the name used in the `CREATE DATABASE` statement.
- Database identifier is recorded when the database is created.
- Time stamp of database creation is also recorded at database creation.
- Names and locations of associated data files and online redo log files are updated when a data file or online redo log file is added to, renamed in, or dropped from the database.
- Tablespace information is updated as tablespaces are added or dropped.
- Online redo log file history is recorded during log switches.
- Location and status of archived logs are recorded when archiving occurs.
- Location and status of backups are recorded by the Recovery Manager (RMAN) utility.
- Current log sequence number is recorded when log switches occur.
- Checkpoint information is recorded as checkpoints are made.

Control File Contents (continued)

The control file consists of two types of sections:

- Reusable
- Not reusable

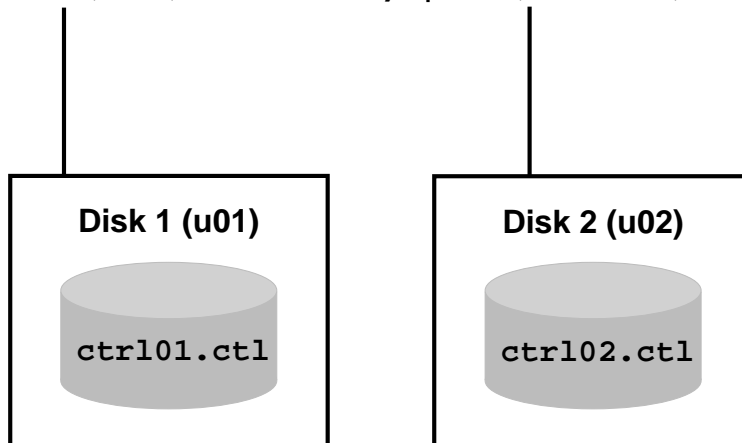
Reusable sections store RMAN information, such as backup data file names and backup online redo log file names. They are used in a circular manner and can be reused only by RMAN.

Note: RMAN is covered in more detail in the course *Oracle9i Database Administration Fundamentals II*.

Multiplexing the Control File

`CONTROL_FILES=`

`$HOME/ORADATA/u01/ctrl01.ctl, $HOME/ORADATA/u02/ctrl02.ctl`



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Multiplexing the Control File

To safeguard against a single point of failure of the control file, it is strongly recommended that the control file be multiplexed, storing each copy on a different physical disk. If a control file is lost, a multiplexed copy of the control file can be used to restart the instance without database recovery.

Control files can be multiplexed up to eight times by:

- Creating multiple control files when the database is created by including the control file names and full path in the initialization parameter file:

```
CONTROL_FILES=$HOME/ORADATA/u01/ctrl01.ctl,  
              $HOME/ORADATA/u02/ctrl02.ctl
```

- Adding a control file after the database is created

Backing Up the Control Files

Because the control file records the physical structure of the database, you should immediately make a backup of your control file after making changes to the physical structure of the database. Backup and recovery of the control file is covered in the course *Oracle9i Database Administration Fundamentals II*.

Multiplexing the Control File When Using SPFILE

1. Alter the SPFILE:

```
ALTER SYSTEM SET control_files =  
'$HOME/ORADATA/u01/ctrl01.ctl',  
'$HOME/ORADATA/u02/ctrl02.ctl' SCOPE=SPFILE;
```

2. Shut down the database:

```
shutdown immediate
```

3. Create additional control files:

```
cp    $HOME/ORADATA/u01/ctrl01.ctl  
      $HOME/ORADATA/u02/ctrl02.ctl
```

4. Start the database:

```
startup
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Multiplexing the Control File When Using SPFILE

Use the following steps to multiplex control files when using an SPFILE.

1. Alter the SPFILE: Using the `ALTER SYSTEM SET` command, alter the SPFILE to include a list of all control files to be used: main control file and multiplexed copies.
2. Shut down the database: Shut down the database in order to create the additional control files on the operating system.
3. Create additional control files: Using the operating system copy command, create the additional control files as required and verify that the files have been created in the appropriate directories.
4. Start the database: When the database is started the SPFILE will be read and the Oracle server will maintain all the control files listed in the `CONTROL_FILES` parameter.

Multiplexing the Control File When Using PFILE

1. Shut down the database:

```
shutdown immediate
```

2. Create additional control files:

```
cp    $HOME/ORADATA/u01/ctrl01.ctl  
      $HOME/ORADATA/u02/ctrl02.ctl
```

3. Add control file names to PFILE:

```
CONTROL_FILES = $HOME/ORADATA/u01/ctrl01.ctl,  
                $HOME/ORADATA/u02/ctrl02.ctl)
```

4. Start the database:

```
startup
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Multiplexing the Control File When Using PFILE

Use the following steps to multiplex control files when using an PFILE.

1. Shut down the database: Shut down the database in order to create the additional control files on the operating system.
2. Create additional control files: Using the operating system copy command, create the additional control files as required and verify that the files have been created in the appropriate directories.
3. Add control file names to PFILE: Alter the PFILE to include a listing of all of the control files.
4. Start the database: When the database is started the PFILE will be read and the Oracle server will maintain all the control files listed in the CONTROL_FILES parameter.

Managing Control Files with OMF

- OMF is created if the `CONTROL_FILES` parameter is not specified.
- Locations are defined by `DB_CREATE_ONLINE_LOG_DEST_n`.
- Names are uniquely generated and displayed in the `alertSID.log`.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Control Files with OMF

Control files are created as OMF automatically during database creation time if the `CONTROL_FILES` parameter is not specified in the initialization parameter file.

If using an `init.ora` file, the `CONTROL_FILES` parameter must be set to the OMF generated names, which can be found by selecting from `V$CONTROLFILE` or from the `alertSID.log`. If an `SPFILE` is used, the `CONTROL_FILES` parameter is automatically set and saved when the database is created.

The locations of the control files are determined by the `DB_CREATE_ONLINE_LOG_DEST_n` parameter. If this parameter is not set, the control files' location will be defined by the `DB_CREATE_FILE_DEST` parameter. If neither of these parameters is set, then the control files will not be of OMF. If the control files are not of OMF, then the `CONTROL_FILES` parameter must be set in the initialization parameter file or an error will be received.

The control file names are uniquely generated (`ora_cm7t30p.ctl`) and displayed in the `alertSID.log` when the files are created.

Obtaining Control File Information

Information about control file status and locations can be retrieved by querying the following views.

- **V\$CONTROLFILE:** Lists the name and status of all control files associated with the instance
- **V\$PARAMETER:** Lists status and location of all parameters
- **V\$CONTROLFILE_RECORD_SECTION:** Provides information about the control file record sections
- **SHOW PARAMETER CONTROL_FILES:** Lists the name, status, and location of the control files

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Obtaining Control File Information

To obtain the location and names of the control files, query the V\$CONTROLFILE view.

```
SQL> SELECT name FROM V$CONTROLFILE;
NAME
-----
/u01/home/db03/ORADATA/u01/ctrl01.ctl
/u01/home/db03/ORADATA/u01/ctrl01.ctl
2 rows selected.
```

You can also query the V\$PARAMETER view.

```
SQL> SELECT name, value from V$PARAMETER
      WHERE name = 'control_files';
NAME                Value
-----
control_files      /u01/home/db03/ORADATA/u01/ctrl01.ctl
```

Obtaining Control File Information (continued)

To obtain information about the different sections of the control files, query the V\$CONTROLFILE_RECORD_SECTION view.

```
SQL> SELECT type, record_size, records_total, records_used
      2 FROM v$controlfile_record_section
      3 WHERE TYPE='DATAFILE' ;
```

TYPE	RECORD_SIZE	RECORDS_TOTAL	RECORDS_USED
DATAFILE	180	40	10

1 row selected.

The RECORDS_TOTAL column specifies the number of records allocated for a special section.

The SHOW PARAMETER command can also be used to find the location of the control files.

```
SQL> SHOW PARAMETER control_files;
```

NAME	TYPE	VALUE
control_files	string	\$HOME/ORADATA/u01/ctrl01.ctl, \$HOME/ORADATA/u02/ctrl02.ctl

Information in several dynamic performance views is obtained from the control file. Below is a list of examples:

- V\$BACKUP
- V\$DATAFILE
- V\$TEMPFILE
- V\$TABLESPACE
- V\$ARCHIVE
- V\$LOG
- V\$LOGFILE
- V\$LOGHIST
- V\$ARCHIVED_LOG
- V\$DATABASE

Obtaining Control File Information (continued)

Using Oracle Enterprise Manager to View Information About Control Files

From the OEM Console:

1. Navigate to Storage > Controlfile
2. Select General tabbed page to view information about control files.

The screenshot displays the Oracle Enterprise Manager (OEM) console interface. On the left, the 'File Navigator' tree shows the hierarchy: Network > Databases > DBA01_STC-SUN01.US.ORACLE > Instance > Storage > Controlfile. The 'Controlfile' node is selected. The main pane shows the 'General' tab. At the top, there are two tabs: 'General' and 'Record Section'. Below the tabs, a table titled 'Controlfile Mirror Images:' lists two entries:

File Name	File Directory
✓ ctrl01.ctl	/home/dba01/ORADATA/u01/
✓ ctrl02.ctl	/home/dba01/ORADATA/u02/

Below the table, the 'Controlfile Information' section displays the following details:

Database ID:	3907540333
Controlfile Type:	CURRENT
Controlfile Creation Date:	07-Jun-2001
Controlfile Sequence Number:	486
Last Change Number:	301001
Date Last Modified:	30-Nov-2001

At the bottom right of the main pane, there are two buttons: 'Show Description' and 'Help'.

Summary

In this lesson, you should have learned how to:

- **Multiplex the control file when using an SPFILE**
- **Multiplex the control file when using an `init.ora`**
- **Manage the control files using OMF**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 6 Overview

This practice covers the following topics:

- **Starting the database without a control file**
- **Multiplexing an existing control file**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 6 Overview

Note: This practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 6: Managing the Control File

- 1 Where is the existing control file located and what is the name?
Hint: Query the V\$CONTROLFILE dynamic performance view.
Note: You can also use V\$PARAMETER, or execute the SHOW PARAMETER command to display the name and the location of the control file.
- 2a Try to start the database without any control files. Simulate this by changing the name of the control file in the parameter file or changing the control file name. What happens?
Hints
 - Connect as user SYS.
 - Shut down the database using the IMMEDIATE option.
 - Using OS command line, copy the control file .ctl as a .bak extension.
 - Remove the control file .ctl.
 - Start the database.
- 2b To resolve the ORA-00205 error:
 - Shut down the database.
 - Rename the copied control file to the appropriate name.
 - Start up the database.
- 3a Multiplex the existing control file, using the u02 directory, and name the new control file ctrl02.ctl. Make sure that the Oracle server is able to write to the new control file. For example, on UNIX use the chmod 660 command.
Hints
 - Before shutting down the database alter the SPFILE (SCOPE=SPFILE) to add the new control file to the initialization file.
 - Shut down the database, and copy the existing control file to a new file with the name ctrl02.ctl in the u02 directory. Use the chmod 660 command on UNIX. **Note:** Normally the permissions on the file would not be changed; this is for the classroom environment.
 - Start up the database.
- b Confirm that the control file was multiplexed and is now being used.
Hint: Query the V\$CONTROLFILE or V\$PARAMETER dynamic performance views, or use the SHOW PARAMETER command to confirm that both control files are being used.
- 4 What is the initial sizing of the data file section in your control file?
Hint: Query the V\$CONTROLFILE_RECORD_SECTION dynamic performance view.



Maintaining Online Redo Log Files

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Explain the purpose of online redo log files**
- **Outline the structure of online redo log files**
- **Control log switches and checkpoints**
- **Multiplex and maintain online redo log files**
- **Manage online redo logs files with OMF**
- **Obtain online redo log file information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Using Online Redo Log Files

Online Redo log files have the following characteristics:

- **Record all changes made to data**
- **Provide a recovery mechanism**
- **Can be organized into groups**
- **At least two groups required**



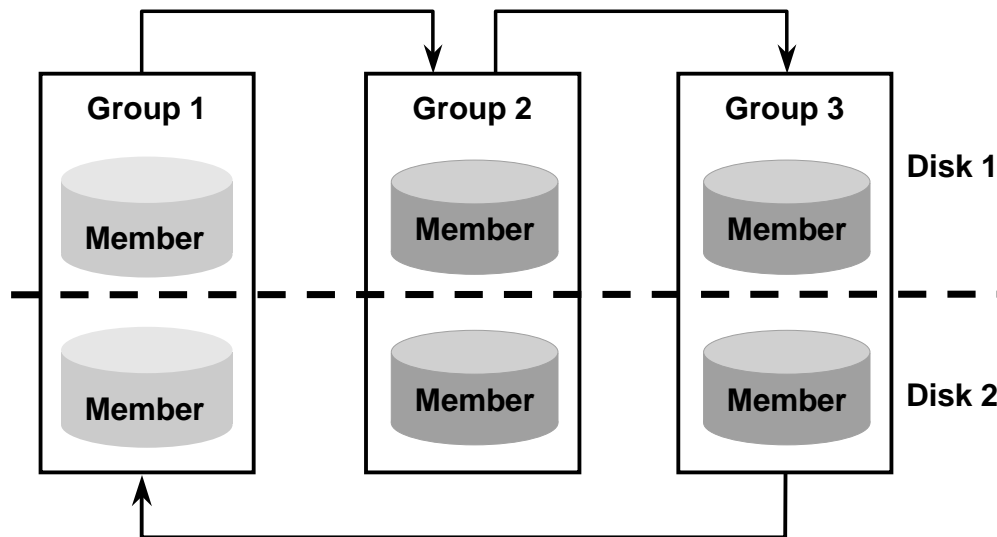
ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Using Online Redo Log Files

Online redo log files provide the means to redo transactions in the event of a database failure. Every transaction is written synchronously to the Redo Log Buffer, then gets flushed to the online redo log files in order to provide a recovery mechanism in case of media failure. (With exceptions such as direct load inserts in objects with the NOLOGGING clause enabled.) This includes transactions that have not yet been committed, undo segment information, and schema and object management statements. Online redo log files are used in a situation such as an instance failure to recover committed data that has not been written to the data files. The online redo log files are used only for recovery.

Structure of Online Redo Log Files



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Structure of Online Redo Log Files

The database administrator can set up the Oracle database to maintain copies of online redo log files to avoid losing database information due to a single point of failure.

Online Redo Log File Groups

- A set of identical copies of online redo log files is called an online redo log file group.
- The LGWR background process concurrently writes the same information to all online redo log files in a group.
- The Oracle server needs a minimum of two online redo log file groups for the normal operation of a database.

Online Redo Log File Members

- Each online redo log file in a group is called a member.
- Each member in a group has identical log sequence numbers and are of the same size. The log sequence number is assigned each time that the Oracle server writes to a log group to uniquely identify each online redo log file. The current log sequence number is stored in the control file and in the header of all data files.

Structure of Online Redo Log Files (continued)

Creating Initial Online Redo Log Files

The initial set of online redo log file groups and members are created during the database creation.

The following parameters limit the number of online redo log files:

- The `MAXLOGFILES` parameter in the `CREATE DATABASE` command specifies the absolute maximum of online redo log file groups.
- The maximum and default value for `MAXLOGFILES` is dependent on your operating system.
- The `MAXLOGMEMBERS` parameter used in the `CREATE DATABASE` command determines the maximum number of members per group. The maximum and default value for `MAXLOGMEMBERS` is dependent on your operating system.

How Online Redo Log Files Work

- **Online Redo log files are used in a cyclic fashion.**
- **When a online redo log file is full, LGWR will move to the next log group.**
 - **Called a log switch**
 - **Checkpoint operation also occurs**
 - **Information written to the control file**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

How Online Redo Log Files Work

The Oracle server sequentially records all changes made to the database in the Redo Log Buffer. The redo entries are written from the Redo Log Buffer to the current online redo log file group by the LGWR process. LGWR writes under the following situations:

- When a transaction commits
- When the Redo Log Buffer becomes one-third full
- When there is more than a megabyte of changed records in the Redo Log Buffer
- Before the DBWn writes modified blocks in the Database Buffer Cache to the data files

Online redo log files are used in a cyclic fashion. Each online redo log file group is identified by a log sequence number that is overwritten each time the log is reused.

Log Switches

LGWR writes to the online redo log files sequentially. When the current online redo log file group is filled, LGWR begins writing to the next group. This is called a log switch.

When the last available online redo log file is filled, LGWR returns to the first online redo log file group and starts writing again.

How Online Redo Log Files Work (continued)

Checkpoints

During a checkpoint:

- DBWn writes a number of dirty database buffers, which are covered by the log that is being checkpointed, to the data files.
- The checkpoint background process CKPT updates the control file to reflect that it has completed a checkpoint successfully. If the checkpoint is caused by a log switch, CKPT also updates the headers of the data files.

Checkpoints can occur for all data files in the database or only for specific data files.

A checkpoint occurs, for example, in the following situations:

- At every log switch
- When an instance has been shut down with the normal, transactional, or immediate option
- When forced by setting the `FAST_START_MTTR_TARGET` initialization parameter
- When manually requested by the database administrator
- When the `ALTER TABLESPACE [OFFLINE NORMAL | READ ONLY | BEGIN BACKUP]` command causes checkpointing on specific data files

Information about each checkpoint is recorded in the `alert_SID.log` file if the `LOG_CHECKPOINTS_TO_ALERT` initialization parameter is set to `TRUE`. The default value of `FALSE` for this parameter does not log checkpoints.

Forcing Log Switches and Checkpoints

- **Forcing a log switch:**

```
ALTER SYSTEM SWITCH LOGFILE;
```

- **Checkpoints can be forced by:**

- **Setting FAST_START_MTTR_TARGET parameter**

```
FAST_START_MTTR_TARGET = 600
```

- **ALTER SYSTEM CHECKPOINT command**

```
ALTER SYSTEM CHECKPOINT;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Forcing Log Switches and Checkpoints

Log switches and checkpoints are automatically performed at certain points in the operation of the database, as identified previously. However, a DBA can force a log switch or a checkpoint to occur.

Forcing Checkpoints

FAST_START_MTTR_TARGET parameter replaces the deprecated parameters:

- FAST_START_IO_TARGET
- LOG_CHECKPOINT_TIMEOUT

These deprecated parameters must not be used if the parameter FAST_START_MTTR_TARGET is used.

In the example above, the FAST_START_MTTR_TARGET parameter has been set so that instance recovery should not take more than 600 seconds. The database will adjust the other parameters to this goal.

Adding Online Redo Log File Groups

```
ALTER DATABASE ADD LOGFILE GROUP 3  
( '$HOME/ORADATA/u01/log3a.rdo',  
  '$HOME/ORADATA/u02/log3b.rdo' )  
SIZE 1M;
```



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Adding Online Redo Log File Groups

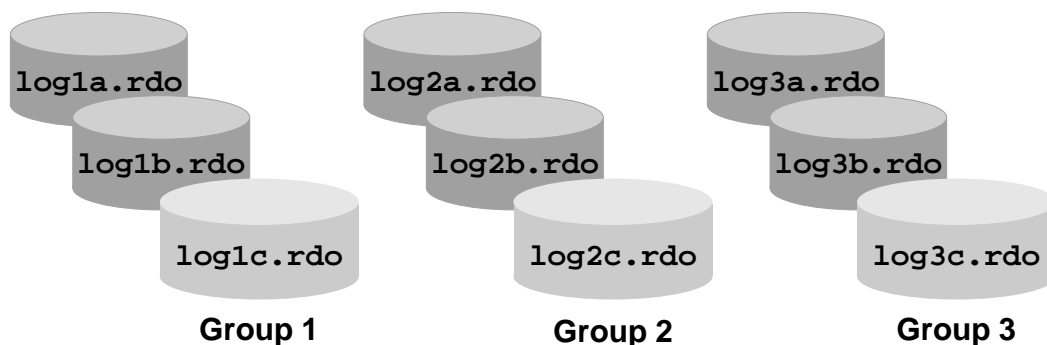
In some cases you might need to create additional log file groups. For example, adding groups can solve availability problems. To create a new group of online redo log files, use the following SQL command:

```
ALTER DATABASE [database]  
  ADD LOGFILE [GROUP integer] filespec  
  [, [GROUP integer] filespec]...
```

You specify the name and location of the members with the file specification. The value of the GROUP parameter can be selected for each online redo log file group. If you omit this parameter, the Oracle server generates its value automatically.

Adding Online Redo Log File Members

```
ALTER DATABASE ADD LOGFILE MEMBER  
'$HOME/ORADATA/u04/log1c.rdo' TO GROUP 1,  
'$HOME/ORADATA/u04/log2c.rdo' TO GROUP 2,  
'$HOME/ORADATA/u04/log3c.rdo' TO GROUP 3;
```



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Adding Online Redo Log File Members

You can add new members to existing online redo log file groups using the following ALTER DATABASE ADD LOGFILE MEMBER command:

```
ALTER DATABASE [database]  
  ADD LOGFILE MEMBER  
    [ 'filename' [REUSE]  
      [, 'filename' [REUSE]]...  
  TO {GROUP integer  
      | ('filename' [, 'filename']...)  
      }  
  ]...
```

Use the fully specified name of the log file members; otherwise the files are created in a default directory of the database server.

If the file already exists, it must have the same size, and you must specify the REUSE option. You can identify the target group either by specifying one or more members of the group or by specifying the group number.

Adding Online Redo Log File Members (continued)

Using Oracle Enterprise Manager to Add Online Redo Log File Groups and Members

From the OEM Console:

1. Navigate to Storage.
2. Click the Redo Log Groups folder.
3. Select Create from the right-mouse menu.
4. Select the General tabbed page.
5. Complete the information to create the online redo log file group and members.
6. Click Create.

Create Redo Log Group - sys@DBA01_STC-SUN01.US....

General

Group #: 3

File Size: 1024 K Bytes

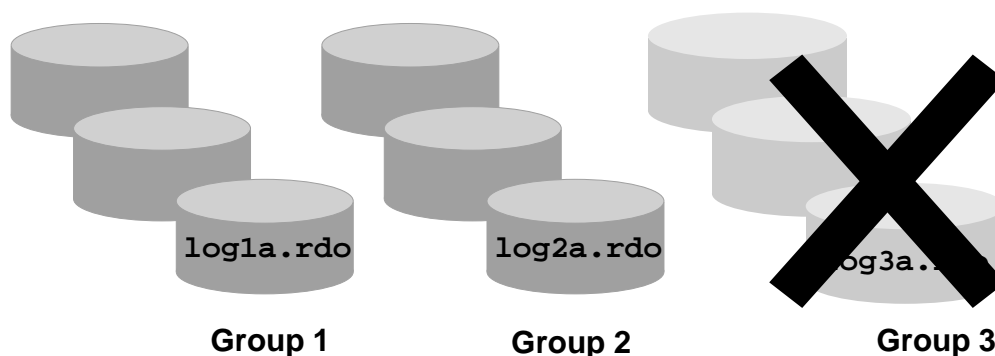
Redo Log Members:

File Name	File Directory
log03a.rdo	/home/dba01/ORADATA/u01/
log03b.rdo	/home/dba01/ORADATA/u02/

Create Cancel Show SQL Help

Dropping Online Redo Log File Groups

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dropping Online Redo Log File Groups

To increase or decrease the size of online redo log file groups, add new online redo log file groups (with the new size) and then drop the old ones.

An entire online redo log file group can be dropped with the following ALTER DATABASE DROP LOGFILE command:

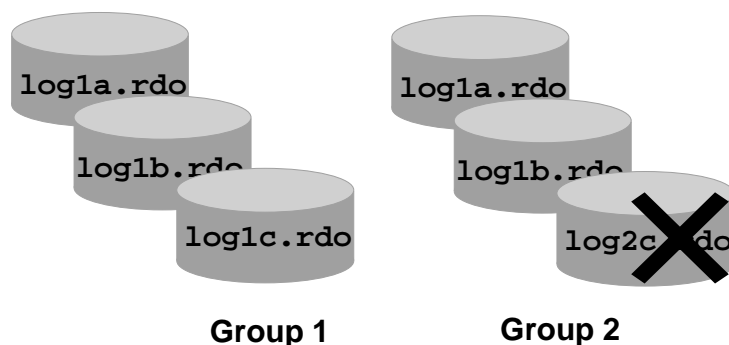
```
ALTER DATABASE [database]
DROP LOGFILE {GROUP integer|('filename'[,
    'filename']...)}
[, {GROUP integer|('filename'[,
    'filename']...)}]...
```

Restrictions

- An instance requires at least two groups of online redo log files.
- An active or current group cannot be dropped.
- When an online redo log file group is dropped, the operating system files are not deleted.

Dropping Online Redo Log File Members

```
ALTER DATABASE DROP LOGFILE MEMBER  
'$HOME/ORADATA/u04/log3c.rdo';
```



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dropping Online Redo Log File Members

You may want to drop an online redo log file member because it is invalid. Use the following ALTER DATABASE DROP LOGFILE MEMBER command if you want to drop one or more specific online redo log file members:

```
ALTER DATABASE [database]  
DROP LOGFILE MEMBER 'filename'[, 'filename']...
```

Restrictions

- If the member you want to drop is the last valid member of the group, you cannot drop that member.
- If the group is current, you must force a log file switch before you can drop the member.
- If the database is running in ARCHIVELOG mode and the log file group to which the member belongs is not archived, then the member cannot be dropped.
- When an online redo log file member is dropped, the operating system file is not deleted if you are not using the OMF feature.

Dropping Online Redo Log File Groups and Members

Using Oracle Enterprise Manager to Drop Online Redo Log File Groups and Members

From the OEM Console:

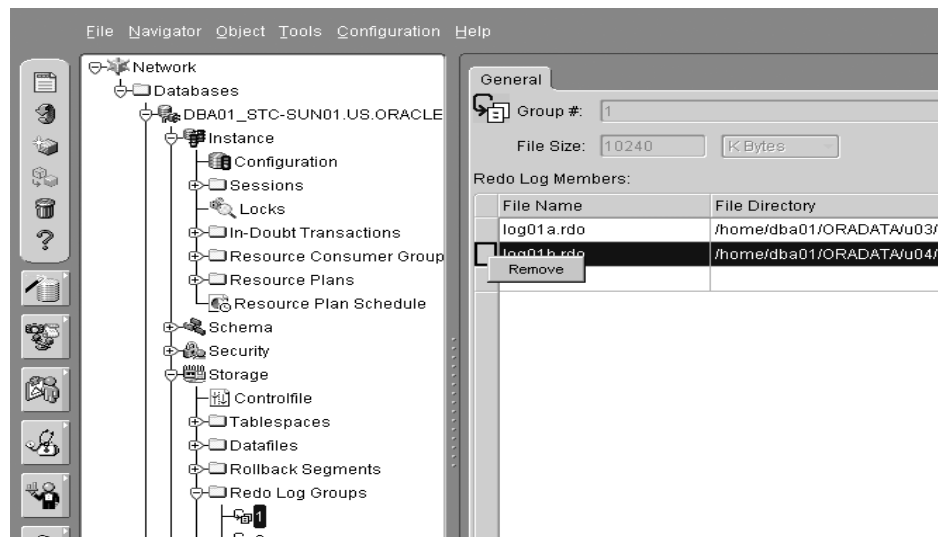
1. Navigate to Storage.

To remove a group:

1. Expand the Redo Log Groups folder and check the status of the online redo log file.
Note: An active or current group cannot be dropped. Two online redo log file groups are required.
2. Select the online redo log file group you want to remove.
3. Select Remove from right-mouse menu.
4. Confirm remove.

To remove a member:

1. Expand the Redo Log Groups folder and navigate to the group containing the member you want to drop.
2. Select the General tabbed page.
3. Highlight the member
4. Select Remove from the right-mouse menu.
5. Confirm remove.



Relocating or Renaming Online Redo Log Files

Relocate or rename online redo log files in one of the two following ways:

- **ALTER DATABASE RENAME FILE command**
 - Shut down the database.
 - Copy the online redo log files to the new location.
 - Place the database in MOUNT mode.
 - Execute the command.
 - Open database for normal operation.

```
ALTER DATABASE RENAME FILE  
  '$HOME/ORADATA/u01/log2a.rdo'  
TO '$HOME/ORADATA/u02/log1c.rdo';
```

- **Add new members and drop old members.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Relocating or Renaming Online Redo Log Files

The locations of online redo log files can be changed by renaming the online redo log files. Before renaming the online redo log files, ensure that the new online redo log file exists. The Oracle server changes only the pointers in the control files, but does not physically rename or create any operating system files.

Steps for Relocating or Renaming Online Redo Log Members

1. Shut down the database.
SQL > SHUTDOWN
2. Copy the online redo log files to the new location.
3. Start up the database and mount, but do not open it.
SQL > CONNECT / as SYSDBA
SQL > STARTUP MOUNT
4. Rename the online redo log members using the ALTER DATABASE RENAME FILE command.
5. Open the database for normal operation.
SQL> ALTER DATABASE OPEN;

Relocating or Renaming Online Redo Log Files (continued)

Using Oracle Enterprise to Relocate or Rename Online Redo Log File Groups and Members

From the OEM Console:

1. Navigate to Storage.
2. Expand Redo Log Groups.
3. Select an online redo log file group.
4. Modify the online redo log file member's File Name or File Directory to rename or relocate members.
5. Click Apply.

Clearing Online Redo Log Files

- **ALTER DATABASE CLEAR LOGFILE** command can be used to reinitialize an online redo log file.

```
ALTER DATABASE CLEAR LOGFILE GROUP 2;
```

- Use the **UNARCHIVED** keyword to avoid archiving the corrupted online redo log file.

```
ALTER DATABASE CLEAR UNARCHIVED  
LOGFILE GROUP 2;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Clearing Online Redo Log Files

An online redo log file might become corrupted while the database is open, and ultimately stop database activity because archiving cannot continue. In this situation the **ALTER DATABASE CLEAR LOGFILE** command can be used to reinitialize the online redo log file without shutting down the database.

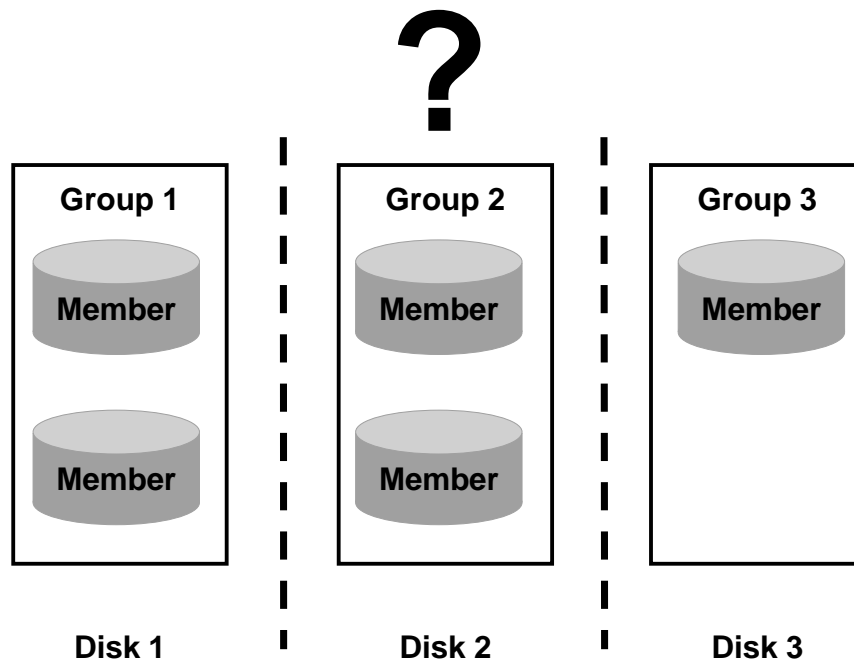
This command can overcome two situations where dropping online redo log files is not possible:

- If there are only two log groups
- The corrupt online redo log file belongs to the current group

If the corrupt online redo log file has not been archived, use the **UNARCHIVED** keyword in the command to clear the corrupted online redo log files and avoids archiving them. The cleared online redo log files are available for use even though they were not archived.

If you clear a log file that is needed for recovery of a backup, then you can no longer recover from that backup. Oracle writes a message in the alert log describing the backups from which you cannot recover.

Online Redo Log File Configuration



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Online Redo Log File Configuration

To determine the appropriate number of online redo log files for a database instance, you must test different configurations.

In some cases, a database instance may require only two groups. In other situations, a database instance may require additional groups to guarantee that the groups are always available to LGWR. For example, if messages in the LGWR trace file or in the alert file indicate that LGWR frequently must wait for a group because a checkpoint has not completed or a group has not been archived, you should add groups.

Although with the Oracle server multiplexed groups can contain different numbers of members, try to build up a symmetric configuration. An asymmetric configuration should be only the temporary result of an unusual situation such as a disk failure.

Location of Online Redo Log Files

When you multiplex the online redo log files, place members of a group on different disks. By doing this, even if one member is not available but other members are available, the instance does not shut down.

Separate archive log files and online redo log files on different disks to reduce contention between the ARCn and LGWR background processes.

Online Redo Log File Configuration (continued)

Data files and online redo log files should be placed on different disks to reduce LGWR and DBWn contention and reduce the risk of losing both data files and online redo log files in the event of media failure.

Sizing Online Redo Log Files

The minimum size of an online redo log file is 50 KB, and the maximum size is specific to the operating system. Members of different groups can have different sizes; however, there is no benefit in having different-sized groups.

Different-sized groups should be required as a temporary result only if you want to change the size of the members of the online redo log file groups. In this case, you should create new online redo log file groups with different sizes, and then remove the old groups.

The following situations might influence the configuration of the online redo log files:

- Number of log switches and checkpoints
- Number and amount of redo entries
- Amount of space on the storage medium; for example, on a tape if archiving is enabled

Managing Online Redo Log Files with OMF

- Define the `DB_CREATE_ONLINE_LOG_DEST_n` parameter:

```
DB_CREATE_ONLINE_LOG_DEST_1  
DB_CREATE_ONLINE_LOG_DEST_2
```

- A group can be added with no file specification:

```
ALTER DATABASE ADD LOGFILE;
```

- Dropping a group:

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Online Redo Log Files with OMF

Define the `DB_CREATE_ONLINE_LOG_DEST_n` parameter: To create online redo log files to be managed by OMF, the `DB_CREATE_ONLINE_LOG_DEST_n` parameter must be defined. The parameter must be set for each multiplexed copy identified by the *n* value. In the example above, two groups have been created with two members each. The names will be generated automatically (such as `ora_1_wo94n2xi.log`) and displayed in the `alertSID.log`. The default size is 100 MB.

To create a new group of online redo log files, the DBA uses the `ALTER DATABASE ADD LOGFILE` command. The command has been modified so that the file specification is not necessary.

The example in the slide adds a log file with two members: One in the location defined by `DB_CREATE_ONLINE_LOG_DEST_1` and one in `DB_CREATE_ONLINE_LOG_DEST_2`. Unique filenames for the log file members are generated automatically and displayed in the `alertSID.log`. The default size is 100 MB.

Dropping a Group

The above example drops the log file Group 3, and its operating system files associated with each OMF log file member in Group 3.

Archived Redo Log Files and OMF

Archived redo log files cannot be OMF.

Obtaining Group and Member Information

Information about a group and its members can be obtained by querying the following views:

- V\$LOG
- V\$LOGFILE

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Obtaining Group and Member Information

V\$LOG view

The following query returns information about the online redo log file from the control file:

```
SQL> SELECT group#, sequence#, bytes, members, status
       2 FROM v$log;
GROUP#      SEQUENCE#      BYTES      MEMBERS      STATUS
-----
1           688           1048576      1           CURRENT
2           689           1048576      1           INACTIVE
2 rows selected.
```

The following items are the most common values for the STATUS column:

- UNUSED: Indicates that the online redo log file group has never been written to. This is the state of an online redo log file that was just added.
- CURRENT: Indicates the current online redo log file group. This implies that the online redo log file group is active.

Obtaining Group and Member Information (continued)

- **ACTIVE:** Indicates that the online redo log file group is active but is not the current online redo log file group. It is needed for crash recovery. It may be in use for block recovery. It may or may not be archived.
- **CLEARING:** Indicates that the log is being re-created as an empty log after an `ALTER DATABASE CLEAR LOGFILE` command. After the log is cleared, the status changes to **UNUSED**.
- **CLEARING_CURRENT:** Indicates that the current log file is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch, such as an input/output (I/O) error writing the new log header.
- **INACTIVE:** Indicates that the online redo log file group is no longer needed for instance recovery. It may or may not be archived.

V\$LOGFILE View

To obtain the names of all the members of a group, query the V\$LOGFILE view.

```
SQL> SELECT member FROM V$LOGFILE;
MEMBER
-----
/u01/home/db03/ORADATA/u03/log02a.rdo
/u01/home/db03/ORADATA/u03/log01a.rdo
```

The value of the **STATUS** column could be one of the following:

- **INVALID:** Indicates that the file is inaccessible
- **STALE:** Indicates that contents of the file are incomplete
- **DELETED:** Indicates that the file is no longer used
- **Blank** indicates that the file is in use

Archived Redo Log Files

- **Filled online redo log files can be archived.**
- **There are two advantages in running the database in ARCHIVELOG mode and archiving online redo log files:**
 - **Recovery:** A database backup together with online and archived redo log files can guarantee recovery of all committed transactions.
 - **Backup:** This can be performed while the database is open.
- **By default, the database is created in NOARCHIVELOG mode.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Archived Redo Log Files

One of the important decisions that a database administrator has to make is whether the database is configured to operate in ARCHIVELOG mode or in NOARCHIVELOG mode.

NOARCHIVELOG Mode

In NOARCHIVELOG mode, the online redo log files are overwritten each time an online redo log file is filled, and log switches occur. LGWR does not overwrite an online redo log file group until the checkpoint for that group is completed.

ARCHIVELOG Mode

If the database is configured to run in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived. Because all changes made to the database are recorded in the online redo log files, the database administrator can use the physical backup and the archived online redo log files to recover the database without losing any committed data.

There are two ways in which online redo log files can be archived:

- Manually
- Automatically (recommended method)

Archived Redo Log Files (continued)

ARCHIVELOG Mode (continued)

The LOG_ARCHIVE_START initialization parameter indicates whether archiving should be automatic or manual when the instance starts up.

- **TRUE:** TRUE indicates that archiving is automatic. ARCn initiates archiving of the filled log group at every log switch.
- **FALSE:** The default value, FALSE indicates that the DBA archives filled online redo log files manually. The DBA must manually execute a command each time you want to archive an online redo log file. All or specific online redo log files can be archived manually.

Archived Redo Log Files

- Accomplished automatically by ARCn
- Accomplished manually through SQL statements
- When successfully archived:
 - An entry in the control file is made
 - Records: archive log name, log sequence number, and high and low system change number (SCN)
- Filled online redo log files cannot be reused until:
 - A checkpoint has taken place
 - File has been archived by ARCn
- Can be multiplexed
- Maintained by the DBA

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Archived Redo Log Files

Information about archived logs can be obtained from V\$INSTANCE.

```
SQL> SELECT archiver
2 FROM v$instance;
ARCHIVE
-----
STOPPED
1 row selected.
```

Note: Archiving is covered in detail in the *Oracle9i Database Administration Fundamentals II* course.

Archived Redo Log Files (continued)

Using Oracle Enterprise Manager to Obtain Archive Information

From OEM Console:

1. Navigate to Instance > Configuration.
2. The General tabbed page identifies:
 - Database and Instance Information–Archive Log Mode: Identifies the mode in which the database is running
 - All Initialization Parameters: Identifies any parameters set for archiving
3. In the Recovery tabbed page you can set and identify the specifics of archiving such as: mode, filename format, and log destinations.

The screenshot shows the Oracle Enterprise Manager console with the 'Recovery' tab selected. The 'Archive Log Mode' section is expanded, showing the following settings:

- ☐ Control instance crash recovery time
- Desired mean time to recover: [] Minutes
- Current estimated mean time to recover: 7 Seconds
- Media Recovery**
- The database is currently in No Archive Log mode.
- ☐ Archive Log Mode
- ☒ Automatic archival
- Log Archive Filename Format: %t_%s.dbf
- It is recommended that archive log files be written to multiple locations spread across different disks.

Archive Log Destination(s)	Status
/oracle/dbs/arch	VALID

i In Archive Log mode, hot backups and recovery to the latest time is possible, but you must provide space for logs. If you change the mode to Archive Log mode, you should take a backup immediately. In No Archive Log mode, only cold backups can be taken, and data can be lost in cases of database corruption.

Summary

In this lesson, you should have learned how to:

- **Explain the use of online redo log files**
- **Obtain online redo log file information**
- **Control log switches and checkpoints**
- **Multiplex and maintain online redo log files**
- **Manage online redo log files with OMF**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 7 Overview

This practice covers the following topics:

- **Creating online redo log file groups and members**
- **Maintaining online redo log file groups and members**
- **Managing online redo log files using OMF**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 7 Overview

Note: This practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 7: Maintaining Online Redo Log Files

- 1 a** List the number and location of existing log files.
Hint: Query the V\$LOGFILE dynamic performance view.
- b** Display the number of online redo log file groups and members your database has.
Hint: Query the V\$LOGFILE dynamic performance view.
- 2** In which database mode is your database configured? Is archiving enabled?
Hints
 - Query the V\$DATABASE dynamic performance view.
 - Query the V\$INSTANCE dynamic performance view.
- 3** Add an online redo log file member to each group in your database located on u04, using the following naming conventions:
Add member to Group 1: log01b.rdo
Add member to Group 2: log02b.rdo
Verify the result.
Hints
 - Execute the ALTER DATABASE ADD LOGFILE MEMBER command to add an online redo log file member to each group.
 - Query the V\$LOGFILE dynamic performance view to verify the result.
- 4** Add an online redo log file group with two members located on u03 and u04 using the following naming conventions and verify the result.
Add Group 3: log03a.rdo and log03b.rdo
Hints
 - Execute the ALTER DATABASE ADD LOGFILE command to create a new group.
 - Query the V\$LOGFILE dynamic performance view to display the name of the new members of the new group.
 - Query the V\$LOG dynamic performance view to display the number of online redo log file groups and members.

Practice 7: Maintaining Online Redo Log Files (continued)

- 5 Remove the online redo log file group created in step 4.

Hints

- Use `ALTER SYSTEM SWITCH LOGFILE` if the log files are active. The number of log switches required will vary. **Note:** Query the database to see which log file is active then decide how many times you need to perform the `ALTER SYSTEM SWITCH LOGFILE` command.
- Execute the `ALTER DATABASE DROP LOGFILE GROUP` command to remove the log group.
- Query the `V$LOG` dynamic performance view to verify the result.
- Remove the operating system files for the group.

- 6 Resize all online redo log files to 1024 KB.

Hints

- You cannot resize log files, therefore, add new logs and drop the old.
- Execute the `ALTER DATABASE ADD LOGFILE GROUP` command to add two new groups with the size 1024 KB.
- Query the `V$LOG` dynamic performance view to check the active group.
- Execute the `ALTER SYSTEM SWITCH LOGFILE` command to force log switches and change the group stage to inactive. The number of log switches required will vary. **Note:** Query the database to see which log file is active, and then decide how many times you should perform the `ALTER SYSTEM SWITCH LOGFILE` command.
- Execute the `ALTER DATABASE DROP LOGFILE` command to remove the unused groups.
- Query the `V$LOG` dynamic performance view to verify the result.

8

Managing Tablespaces and Data Files

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Define the purpose of tablespaces and data files**
- **Create tablespaces**
- **Manage tablespaces**
- **Create and manage tablespaces using Oracle Managed Files (OMF)**
- **Obtain tablespace information**

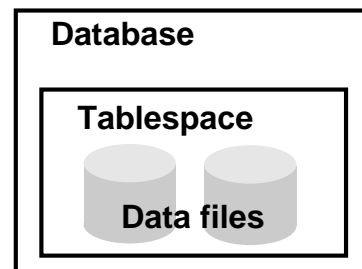
ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Tablespaces and Data Files

Oracle stores data logically in tablespaces and physically in data files.

- **Tablespaces:**
 - Can belong to only one database at a time
 - Consist of one or more data files
 - Are further divided into logical units of storage
- **Data files:**
 - Can belong to only one tablespace and one database
 - Are a repository for schema object data



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Tablespaces and Data Files

Databases, tablespaces, and data files are closely related, but they have important differences:

- An Oracle database consists of one or more logical storage units called tablespaces, which collectively store all of the database's data.
- Each tablespace in an Oracle database consists of one or more files called data files, which are physical structures that conform with the operating system in which Oracle is running.
- A database's data is collectively stored in the data files that constitute each tablespace of the database. For example, the simplest Oracle database would have one tablespace and one data file. Another database can have three tablespaces, each consisting of two data files (for a total of six data files).

Types of Tablespaces

- **SYSTEM tablespace**
 - Created with the database
 - Contains the data dictionary
 - Contains the **SYSTEM** undo segment
- **Non-SYSTEM tablespace**
 - Separate segments
 - Eases space administration
 - Controls amount of space allocated to a user

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Types of Tablespaces

The DBA creates tablespaces for increased control and ease of maintenance. The Oracle server perceives two types of tablespaces: **SYSTEM** and all others.

SYSTEM tablespace

- Created with the database
- Required in all databases
- Contains the data dictionary, including stored program units
- Contains the **SYSTEM** undo segment
- Should not contain user data, although it is allowed

Non-SYSTEM tablespaces

- Enable more flexibility in database administration
- Separate undo, temporary, application data, and application index segments
- Separate data by backup requirements
- Separate dynamic and static data
- Control the amount of space allocated to the user's objects

Creating Tablespaces

A tablespace is created using the command:

CREATE TABLESPACE

```
CREATE TABLESPACE userdata  
DATAFILE '/u01/oradata/userdata01.dbf' SIZE 5M;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating Tablespaces

You create a tablespace with the CREATE TABLESPACE command:

```
CREATE TABLESPACE tablespace  
  [DATAFILE clause]  
  [MINIMUM EXTENT integer[K|M]]  
  [BLOCKSIZE integer [K]]  
  [LOGGING|NOLOGGING]  
  [DEFAULT storage_clause ]  
  [ONLINE|OFFLINE]  
  [PERMANENT|TEMPORARY]  
  [extent_management_clause]  
  [segment_management_clause]
```

Creating Tablespaces (continued)

Where:

Tablespace: This is the name of the tablespace to be created.

DATAFILE: This specifies the data file or data files that make up the tablespace.

MINIMUM EXTENT: This ensures that every used extent size in the tablespace is a multiple of the integer. Use K or M to specify this size in kilobytes or megabytes.

BLOCKSIZE: Specifies a nonstandard block size for the tablespace. In order to specify this clause, you must have the `DB_CACHE_SIZE` and at least one `DB_nK_CACHE_SIZE` parameter set, and the integer you specify in this clause must correspond with the setting of one `DB_nK_CACHE_SIZE` parameter setting.

LOGGING: This specifies that, by default, all tables, indexes, and partitions within the tablespace have all changes written to online redo log files. `LOGGING` is the default.

NOLOGGING: This specifies that, by default, all tables, indexes, and partitions within the tablespace do not have all changes written to online redo log files. `NOLOGGING` affects only some DML and DDL commands, for example, direct loads.

DEFAULT: Specifies the default storage parameters for all objects created in the tablespace creation.

OFFLINE: This makes the tablespace unavailable immediately after creation.

PERMANENT: This specifies that the tablespace can be used to hold permanent objects.

TEMPORARY: This specifies that the tablespace be used only to hold temporary objects; for example, segments used by implicit sorts caused by an `ORDER BY` clause. It cannot specify `EXTENT MANAGEMENT LOCAL` or the `BLOCKSIZE` clause.

extent_management_clause: This clause specifies how the extents of the tablespace are managed. This clause is discussed in a subsequent section of this lesson.

segment_management_clause: This is relevant only for permanent, locally managed tablespaces. It lets you specify whether Oracle should track the used and free space in the segments in the tablespace using free lists or bitmaps.

**datafile_clause ::= filename [SIZE integer[K|M] [REUSE]
[autoextend_clause]**

filename: This is the name of a data file in the tablespace.

SIZE: This specifies the size of the file. Use K or M to specify the size in kilobytes or megabytes.

REUSE: This allows the Oracle server to reuse an existing file.

autoextend_clause: This clause enables or disables automatic extension of the data file.

NEXT: This specifies the size in bytes of the next increment of disk space to be allocated automatically when more extents are required.

Creating Tablespaces (continued)

Where:

MAXSIZE: This specifies the maximum disk space allowed for automatic extension of a data file.

UNLIMITED: This specifies the disk space that can be allocated to the data file, or that the tempfile is not limited.

See also: *Oracle9i SQL Reference* and *Oracle9i Concepts* for additional information.

Creating Tablespaces (continued)

Using Oracle Enterprise Manager to Create a Tablespace

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select Create from the right-mouse menu.
3. Complete the information in the General and Storage tabbed pages as required for the tablespace.
4. Click Create.

The screenshot shows the 'Storage' tab of the Oracle Enterprise Manager console. The 'Name' field is set to 'USERDATA'. Below it, the 'Datafiles' section contains a table with one row:

File Name	File Directory	Size	
USERDATA01.dbf	/home/dba01/ORADATA/u01/	100	MB

Below the table, there are icons for adding and deleting datafiles. The 'Status' section has radio buttons for 'Online' (selected) and 'Offline', and a 'Read Only' checkbox. The 'Type' section has radio buttons for 'Permanent' (selected) and 'Temporary', and a checkbox for 'Set as Default Temporary Tablespace'. At the bottom, there are buttons for 'Create', 'Cancel', 'Show SQL', and 'Help'.

Space Management in Tablespaces

- **Locally managed tablespace:**
 - Free extents are managed in the tablespace.
 - Bitmap is used to record free extents.
 - Each bit corresponds to a block or group of blocks.
 - Bit value indicates free or used.
- **Dictionary-managed tablespace:**
 - Free extents are managed by the data dictionary.
 - Appropriate tables are updated when extents are allocated or deallocated.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Space Management in Tablespaces

Tablespaces allocate space in extents. Tablespaces can be created to use one of the following two different methods of keeping track of free and used space:

Locally managed tablespaces: The extents are managed within the tablespace via bitmaps. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, the Oracle server changes the bitmap values to show the new status of the blocks. Locally managed is the default beginning with Oracle9i.

Dictionary-managed tablespaces: The extents are managed by the data dictionary. The Oracle server updates the appropriate tables in the data dictionary whenever an extent is allocated or deallocated.

Locally Managed Tablespaces

- Reduced contention on data dictionary tables
- No undo generated when space allocation or deallocation occurs
- No coalescing required

```
CREATE TABLESPACE userdata
DATAFILE '/u01/oradata/userdata01.dbf' SIZE 500M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Locally Managed Tablespaces

The `LOCAL` option of the `EXTENT MANAGEMENT` clause specifies that a tablespace is to be locally managed. By default a tablespace is locally managed.

`extent_management_clause`:

```
[ EXTENT MANAGEMENT [ DICTIONARY | LOCAL
[ AUTOALLOCATE | UNIFORM [SIZE integer[K|M]] ] ] ]
```

where:

DICTIONARY: Specifies that the tablespace is managed using dictionary tables

LOCAL: Specifies that the tablespace is locally managed via bitmaps. If you specify `LOCAL`, you cannot specify `DEFAULT storage_clause`, `MINIMUM EXTENT`, or `TEMPORARY`.

AUTOALLOCATE: Specifies that the tablespace is system managed. Users cannot specify an extent size. This is the default.

UNIFORM: Specifies that the tablespace is managed with uniform extents of `SIZE` bytes. Use `K` or `M` to specify the extent size in kilobytes or megabytes. The default size is 1 MB.

Locally Managed Tablespaces (continued)

The `EXTENT MANAGEMENT` clause can be used in various `CREATE` commands:

- For a permanent tablespace, you can specify `EXTENT MANAGEMENT LOCAL` in the `CREATE TABLESPACE` command.

Note: Prior to Oracle9i Database Release 2, the `SYSTEM` tablespace was not locally managed.

- For a temporary tablespace, you can specify `EXTENT MANAGEMENT LOCAL` in the `CREATE TEMPORARY TABLESPACE` command.

Advantages of Locally Managed Tablespaces

Locally managed tablespaces have the following advantages over dictionary-managed tablespaces:

- Local management avoids recursive space management operations. This can occur in dictionary-managed tablespaces if consuming or releasing space in an extent results in another operation that consumes or releases space in an undo segment or data dictionary table.
- Because locally managed tablespaces do not record free space in data dictionary tables, they reduce contention on these tables.
- Local management of extents automatically tracks adjacent free space, eliminating the need to coalesce free extents.
- The sizes of extents that are managed locally can be determined automatically by the system.
- Changes to the extent bitmaps do not generate undo information because they do not update tables in the data dictionary (except for special cases such as tablespace quota information).

Dictionary-Managed Tablespaces

- Extents are managed in the data dictionary.
- Each segment stored in the tablespace can have a different storage clause.
- Coalescing is required.

```
CREATE TABLESPACE userdata  
DATAFILE '/u01/oradata/userdata01.dbf'  
SIZE 500M EXTENT MANAGEMENT DICTIONARY  
DEFAULT STORAGE  
(initial 1M NEXT 1M PCTINCREASE 0);
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dictionary-Managed Tablespaces

Segments in dictionary-managed tablespaces can have a customized storage. This storage is more flexible than locally managed tablespaces but much less efficient.

Migrating a Dictionary-Managed SYSTEM Tablespace

Migrate a dictionary managed SYSTEM tablespace to locally managed:

```
DBMS_SPACE_ADMIN.  
TABLESPACE_MIGRATE_TO_LOCAL( 'SYSTEM' );
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Migrating a Dictionary-Managed SYSTEM Tablespace

Use the following steps to convert your SYSTEM tablespace from dictionary managed to locally managed:

1. Make a complete backup of your database.
2. Ensure that the database has a default temporary tablespace that is not SYSTEM. The temporary tablespace is created using the CREATE TEMPORARY TABLESPACE command.
3. Eliminate any undo (rollback) segments in dictionary-managed tablespaces.
4. There should be at least one online undo segment in a locally managed tablespace, or an undo tablespace should be online.
5. All tablespaces other than the tablespace containing the undo space and the default temporary tablespace should be placed in READ ONLY mode.
6. Start up the instance in restricted mode.
7. Migrate the SYSTEM tablespace by using:

```
DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL( 'SYSTEM' );
```

Note: Any non-SYSTEM dictionary-managed tablespace that is not migrated to locally managed prior to migrating the SYSTEM tablespace cannot be altered to READ WRITE. The tablespace will remain READ ONLY. Only locally managed tablespaces can be altered to READ WRITE after the migration of the SYSTEM tablespace.

Undo Tablespace

- Used to store undo segments
- Cannot contain any other objects
- Extents are locally managed
- Can only use the **DATAFILE** and **EXTENT MANAGEMENT** clauses

```
CREATE UNDO TABLESPACE undo1  
DATAFILE '/u01/oradata/undo01.dbf' SIZE 40M;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Undo Tablespace

An undo tablespace is used with Automatic Undo Management. Refer to the “Managing Undo Data” lesson for more information about Automatic Undo Management.

```
CREATE UNDO TABLESPACE tablespace [DATAFILE clause]
```

Temporary Tablespaces

- Used for sort operations
- Can be shared by multiple users
- Cannot contain any permanent objects
- Locally managed extents recommended

```
CREATE TEMPORARY TABLESPACE temp  
TEMPFILE '/u01/oradata/temp01.dbf' SIZE 20M  
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Temporary Tablespaces

You can manage space for sort operations more efficiently by designating temporary tablespaces exclusively for sort segments. No permanent schema objects can reside in a temporary tablespace.

Sort, or temporary, segments are used when a segment is shared by multiple sort operations. Temporary tablespaces provide performance improvements when you have multiple sorts that are too large to fit into memory. The sort segment of a given temporary tablespace is created at the time of the first sort operation of the instance. The sort segment expands by allocating extents until the segment size is equal to or greater than the total storage demands of all of the active sorts running on that instance.

Note: Nonstandard block sizes cannot be specified when creating temporary tablespaces.

Temporary Tablespaces (continued)

Locally managed temporary tablespaces have temporary data files (temp files), which are similar to ordinary data files except that:

- Tempfiles are always set to the NOLOGGING mode.
- You cannot make a tempfile read-only.
- You cannot rename a temp file.
- You cannot create a tempfile with the ALTER DATABASE command.
- Temp files are required for read-only databases.
- Media recovery does not recover tempfiles.

To optimize the performance of a sort in a temporary tablespace, set the UNIFORM SIZE to be a multiple of the parameter SORT_AREA_SIZE.

Temporary Tablespaces (continued)

Using Oracle Enterprise Manager to Create a Temporary Tablespace

From OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select Create from the right-mouse menu.
3. Enter details in the General tabbed page.
4. Select the Temporary option in the Type region on the General tabbed page.
5. Click the Storage tabbed page, and enter the storage information.
6. Click Create.

Create Tablespace - system@U621

General Storage

Name: TEMP

Tempfiles

	File Name	File Directory	Size	
↑	TEMP.dbf	/home1/user621/O...	5	MB

Status

☒ Online ☐ Read Only

☐ Offline

Type

☐ Permanent

☒ Temporary

☐ Set as Default Temporary Tablespace

☐ Undo

Create Cancel Show SQL Help

Default Temporary Tablespace

- **Specifies a database-wide default temporary tablespace**
- **Eliminates using `SYSTEM` tablespace for storing temporary data**
- **Can be created by using:**
 - `CREATE DATABASE`
 - `ALTER DATABASE`

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Default Temporary Tablespace

A default temporary tablespace can be created and set by:

- Using the `CREATE TABLESPACE` command to create a temporary tablespace
- Using the `ALTER DATABASE` command

When creating a database without a default temporary tablespace the `SYSTEM` tablespace is assigned to any user created without a `TEMPORARY TABLESPACE` clause. Also a warning is placed in the `alert_sid.log` stating that the `SYSTEM` tablespace is the default temporary tablespace. Creating a default temporary tablespace during database creation prevents the `SYSTEM` tablespace from being used for temporary space.

After database creation, a default temporary tablespace can be set by creating a temporary tablespace and then altering the database.

```
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp;
```

Once defined, users who are not explicitly assigned to a temporary tablespace are assigned to the default temporary tablespace.

The default temporary tablespace can be changed at any time by using the `ALTER DATABASE DEFAULT TEMPORARY TABLESPACE` command. When the default temporary tablespace is changed, all users assigned the default temporary tablespace are reassigned to the new default.

Creating a Default Temporary Tablespace

- During database creation:

```
CREATE DATABASE DBA01
LOGFILE
GROUP 1 ( '$HOME/ORADATA/u01/redo01.log' ) SIZE 100M,
GROUP 2 ( '$HOME/ORADATA/u02/redo02.log' ) SIZE 100M,
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
MAXDATAFILES 100
MAXINSTANCES 1
DATAFILE '$HOME/ORADATA/u01/system01.dbf' SIZE 325M
UNDO TABLESPACE undotbs
DATAFILE '$HOME/ORADATA/u02/undotbs01.dbf' SIZE 200
DEFAULT TEMPORARY TABLESPACE temp
TEMPFILE '$HOME/ORADATA/u03/temp01.dbf' SIZE 4M
CHARACTER SET US7ASCII
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Default Temporary Tablespace

- After database creation:

```
ALTER DATABASE  
DEFAULT TEMPORARY TABLESPACE default_temp2;
```

- To find the default temporary tablespace for the database query `DATABASE_PROPERTIES`:

```
SELECT * FROM DATABASE_PROPERTIES;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Default Temporary Tablespace (continued)

Using Oracle Enterprise Manager to Create a Default Temporary Tablespace

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select Create from the right-mouse menu.
3. Select Temporary and Set as Default Temporary Tablespace in the General tabbed page.
4. Complete necessary information in the Storage tabbed page.
5. Click Create.

Create Tablespace - system@U621

General Storage

Name: TEMP

Tempfiles

File Name	File Directory	Size	Unit
TEMP.dbf	/home1/user621/O...	5	MB

Status

☒ Online ☐ Read Only

☐ Offline Normal

Type

☐ Permanent

☒ Temporary

☒ Set as Default Temporary Tablespace

☐ Undo

Create Cancel Show SQL Help

Restrictions on Default Temporary Tablespace

Default temporary tablespaces cannot be:

- **Dropped until after a new default is made available**
- **Taken offline**
- **Altered to a permanent tablespace**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Restrictions on Default Temporary Tablespace

Dropping a Default Temporary Tablespace

You cannot drop the default temporary tablespace until after a new default is made available. The `ALTER DATABASE` command must be used to change the default temporary tablespace to a new default. The old default temporary tablespace is then dropped only after a new default temporary tablespace is made available. Users assigned to the old default temporary tablespace are automatically reassigned to the new default temporary tablespace.

Changing the Type of a Default Temporary Tablespace

Because a default temporary tablespace must be either the `SYSTEM` tablespace or a temporary tablespace, you cannot change the default temporary tablespace to a permanent type.

Taking Default Temporary Tablespace Offline

Tablespaces are taken offline to make that part of the database unavailable to other users (for example, an offline backup, maintenance, or making a change to an application that uses the tablespace). Because none of these situations applies to a temporary tablespace, you cannot take a default temporary tablespace offline.

Read-Only Tablespaces

- Use the following command to place a tablespace in read-only mode:

```
ALTER TABLESPACE userdata READ ONLY;
```

- Causes a checkpoint
- Data available only for read operations
- Objects can be dropped from tablespace

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Read-Only Tablespaces

The `ALTER TABLESPACE [tablespace] READ ONLY` command places the tablespace in a transitional read-only mode. In this transitional state, no further write operations can occur in the tablespace except for the rollback of existing transactions that previously modified blocks in the tablespace. After all of the existing transactions have been either committed or rolled back, the read-only command completes, and the tablespace is placed in read-only mode.

You can drop items, such as tables and indexes, from a read-only tablespace, because these commands affect only the data dictionary. This is possible because the `DROP` command updates only the data dictionary, but not the physical files that make up the tablespace. For locally managed tablespaces, the dropped segment is changed to a temporary segment, to prevent the bitmap from being updated. To make a read-only tablespace writable, all of the data files in the tablespace must be online. Making tablespaces read-only causes a checkpoint on the data files of the tablespace.

Read-Only Tablespaces (continued)

Making tablespaces read-only prevents further write operations on the data files in the tablespace. Therefore, the data files can reside on read-only media, such as CD-ROMs or write-once (WORM) drives. Read-only tablespaces eliminate the need to perform backups of large, static portions of a database.

Read-Only Tablespaces (continued)

Using Oracle Enterprise Manager to Make a Tablespace Read-Only

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select the tablespace.
3. Select the Read Only check box in the Status region of the General tabbed page.
4. Click Apply.

The screenshot shows the Oracle Enterprise Manager console interface for managing a tablespace. The 'Storage' tab is active, and the tablespace name is 'USERDATA'. Below the 'Datafiles' table, the 'Status' section is visible, with 'Read Only' selected. The 'Type' section shows 'Permanent' selected. At the bottom, there are buttons for 'Apply', 'Revert', 'Show SQL', and 'Help'.

File Name	File Directory	Size	Use
USERDATA.dbf	/home1/user621/O...	5 MB	0.0

Status

☐ Online ☒ Read Only

☐ Offline

Type

☐ Permanent

☐ Temporary

☐ Set as Default Temporary Tablespace

☐ Undo

Apply Revert Show SQL Help

Taking a Tablespace Offline

- **Not available for data access**
- **Tablespaces that cannot be taken offline:**
 - **SYSTEM** tablespace
 - **Tablespaces with active undo segments**
 - **Default temporary tablespace**
- **To take a tablespace offline:**

```
ALTER TABLESPACE userdata OFFLINE;
```

- **To bring a tablespace online:**

```
ALTER TABLESPACE userdata ONLINE;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Taking a Tablespace Offline

A tablespace is normally online so that the data contained within it is available to database users. However, the database administrator might take a tablespace offline to:

- Make a portion of the database unavailable, while allowing normal access to the remainder of the database
- Perform an offline tablespace backup (although a tablespace can be backed up while online and in use)
- Recover a tablespace or data file while the database is open
- Move a data file while the database is open

Offline Status of a Tablespace

When a tablespace goes offline, Oracle does not permit any subsequent SQL statements to reference objects contained in that tablespace. Users trying to access objects in a tablespace that is offline receive an error.

When a tablespace goes offline or comes back online, the event is recorded in the data dictionary and in the control file. If a tablespace is offline when you shut down a database, the tablespace remains offline and is not checked when the database is subsequently mounted and reopened.

Taking a Tablespace Offline (continued)

The Oracle instance automatically switches a tablespace from online to offline when certain errors are encountered (for example, when the database writer process, DBWn, fails in several attempts to write to a data file of the tablespace). The different error situations are covered in more detail in the course *Oracle9i Database Administration Fundamentals II*.

Taking Tablespaces Offline

Whenever the database is open, a database administrator can take any tablespace, except the SYSTEM tablespace or any tablespace with active undo segments or temporary segments, offline. When a tablespace is taken offline, the Oracle server takes all the associated data files offline.

```
ALTER TABLESPACE tablespace
    { ONLINE | OFFLINE [ NORMAL | TEMPORARY | IMMEDIATE | FOR RECOVER ] }
```

Where:

NORMAL: Flushes all blocks in all data files in the tablespace out of the SGA. This is the default. You need not perform media recovery on this tablespace before bringing it back online. Use the NORMAL clause whenever possible.

TEMPORARY: Performs a checkpoint for all online data files in the tablespace even if some files could not be written. Any offline files may require media recovery.

IMMEDIATE: Does not ensure that tablespace files are available and does not perform a checkpoint. You must perform media recovery on the tablespace before bringing it back online.

FOR RECOVER: Takes tablespaces offline for tablespace point-in-time recovery.

Taking a Tablespace Offline (continued)

Using Oracle Enterprise Manager to Take a Tablespace Offline

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select the tablespace.
3. Select Offline in the Status region of the General tabbed page.
4. Select the Mode from the drop-down menu.
5. Click Apply.

The screenshot shows the Oracle Enterprise Manager console interface for managing a tablespace. The 'Storage' tab is active, and the tablespace name is 'USERDATA'. The 'Datafiles' section contains a table with the following data:

File Name	File Directory	Size	Use
USERDATA.dbf	/home1/user621/O...	5 MB	0.0

Below the table, there are icons for editing and deleting. The 'Status' section has two radio buttons: 'Online' and 'Offline', with 'Offline' selected. A 'Read Only' checkbox is also present. A dropdown menu shows 'Normal'. The 'Type' section has three radio buttons: 'Permanent', 'Temporary', and 'Undo', with 'Permanent' selected. There is also a checkbox for 'Set as Default Temporary Tablespace'. At the bottom, there are buttons for 'Apply', 'Revert', 'Show SQL', and 'Help'.

Changing Storage Settings

- Using ALTER TABLESPACE command to change storage settings:

```
ALTER TABLESPACE userdata MINIMUM EXTENT 2M;
```

```
ALTER TABLESPACE userdata  
DEFAULT STORAGE (INITIAL 2M NEXT 2M  
MAXEXTENTS 999);
```

- Storage settings for locally managed tablespaces cannot be altered.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Changing Storage Settings

Use the ALTER TABLESPACE command to alter the default storage definition of a tablespace:

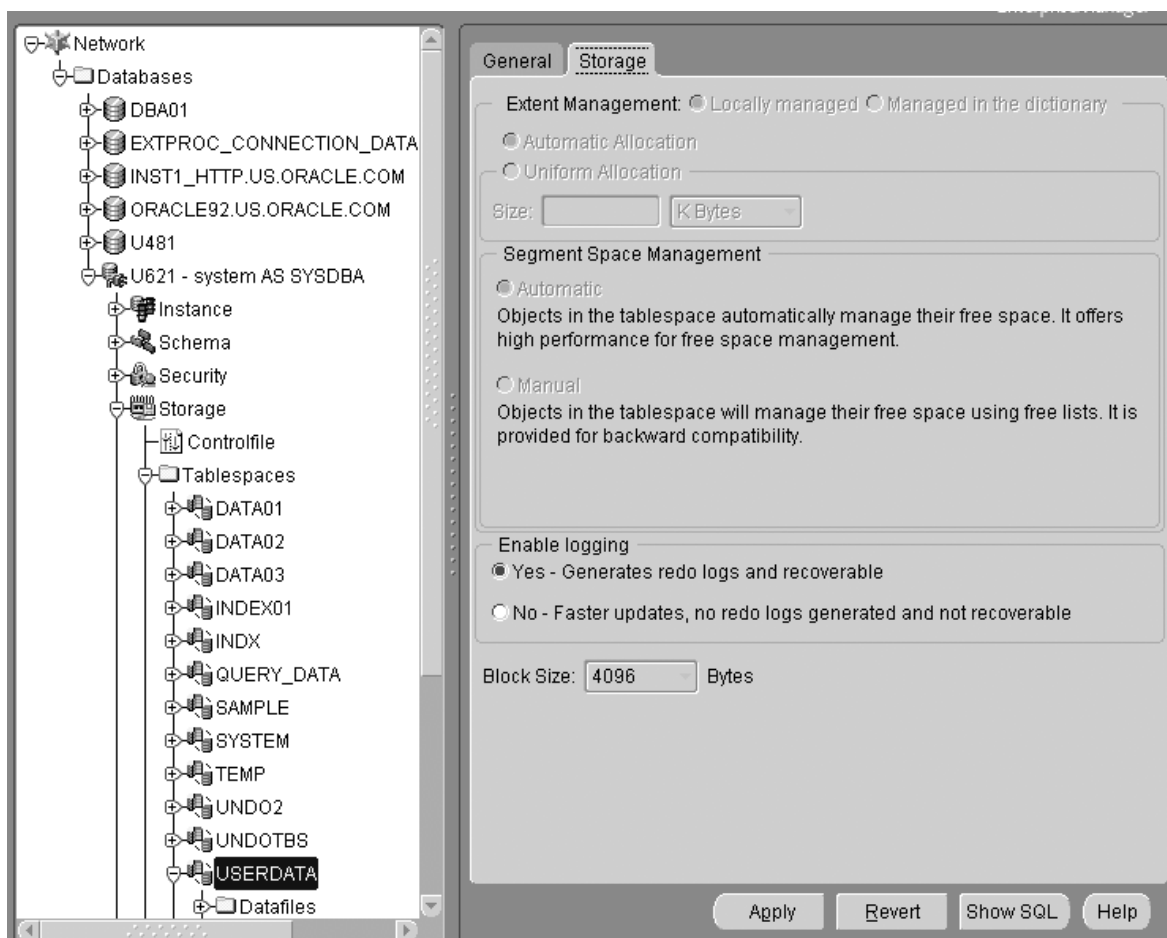
```
ALTER TABLESPACE tablespace  
[MINIMUM EXTENT integer[K|M]  
|DEFAULT storage_clause ]
```

Changing Storage Settings (continued)

Using Oracle Enterprise Manager to Change the Storage Settings

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Right-click the tablespace.
3. Select View/Edit Details from the menu.
4. Click the Storage tabbed page and make necessary changes.
5. Click Apply.



Resizing a Tablespace

A tablespace can be resized by:

- **Changing the size of a data file:**
 - Automatically using **AUTOEXTEND**
 - Manually using **ALTER DATABASE**
- **Adding a data file using **ALTER TABLESPACE****

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Resizing a Tablespace

You can enlarge a tablespace by:

- Change the size of a data file:
 - Automatically during tablespace creation

```
SQL> CREATE TABLESPACE userdata02
  2  DATAFILE '/u01/oradata/userdata02.dbf' SIZE 5M
  3  AUTOEXTEND ON NEXT 2M MAXSIZE 200M;
```
 - By specifying **AUTOEXTEND** after tablespace creation

```
SQL> ALTER DATABASE
  2  DATAFILE '/u01/oradata/userdata02.dbf '
  3  AUTOEXTEND ON NEXT 2M;
```
 - Manually

```
SQL> ALTER DATABASE
  2  DATAFILE '/u01/oradata/userdata02.dbf' RESIZE 5M;
```

Resizing a Tablespace (continued)

- Add a data file to a tablespace.

```
SQL> ALTER TABLESPACE userdata02  
2  ADD DATAFILE '/u01/oradata/userdata03.dbf'  
3  SIZE 5M;
```

Enabling Automatic Extension of Data Files

- Can be resized automatically with the following commands:

- CREATE DATABASE
- CREATE TABLESPACE
- ALTER TABLESPACE ... ADD DATAFILE

- Example:

```
CREATE TABLESPACE user_data
DATAFILE
  '/u01/oradata/userdata01.dbf' SIZE 200M
  AUTOEXTEND ON NEXT 10M MAXSIZE 500M;
```

- Query the DBA_DATA_FILES view to determine whether AUTOEXTEND is enabled.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Enabling Automatic Extension of Data Files

Specifying AUTOEXTEND for a New Data File

The AUTOEXTEND clause enables or disables the automatic extension of data files. Files increase in specified increments up to a specified maximum.

Benefits of using the AUTOEXTEND clause:

- Reduces need for immediate intervention when a tablespace runs out of space
- Ensures applications will not halt because of failures to allocate extents

When a data file is created, the following SQL commands can be used to enable automatic extension of the data file:

- CREATE DATABASE
- CREATE TABLESPACE ... DATAFILE
- ALTER TABLESPACE ... ADD DATAFILE

Enabling Automatic Extension of Data Files (continued)

Specifying AUTOEXTEND for a New Data File (continued)

Use the ALTER DATABASE command to modify a data file and enable automatic extension:

```
ALTER DATABASE DATAFILE filespec [autoextend_clause]
autoextend_clause ::= [ AUTOEXTEND { OFF|ON[NEXT integer[K|M]]
                        [MAXSIZE UNLIMITED | integer[K|M]] } ]
```

where:

AUTOEXTEND OFF: Disables the automatic extension of the data file

AUTOEXTEND ON: Enables the automatic extension of the data file

NEXT: Specifies the disk space to allocate to the data file when more extents are required

MAXSIZE: Specifies the maximum disk space allowed for allocation to the data file

UNLIMITED: Sets no limit on allocating disk space to the data file

Specifying AUTOEXTEND for an Existing Data File

Use the SQL command ALTER DATABASE to enable or disable automatic file extension for existing data files:

```
ALTER DATABASE [database]
DATAFILE 'filename'[, 'filename']... autoextend_clause
```

Determining Whether AUTOEXTEND is Enabled or Disabled

Query DBA_DATA_FILES view to determine whether AUTOEXTEND is enabled and examine the AUTOEXTENSIBLE column.

```
SQL> SELECT tablespace_name, file_name, autoextensible
       2 FROM dba_data_files;
```

TABLESPACE_NAME	FILE_NAME	AUTOEXTENSIBLE
-----	-----	-----
SYSTEM	/home/dba01/ORADATA/u01/system01.dbf	YES
DATA01	/home/dba01/ORADATA/u04/data01.dbf	NO
USERS	/home/dba01/ORADATA/u03/users01.dbf	NO
INDX	/home/dba01/ORADATA/u06/indx01.dbf	NO
SAMPLE	/home/dba01/ORADATA/u02/sample01.dbf	YES
DATA02	/home/dba01/ORADATA/u03/data02.dbf	NO
INDEX01	/home/dba01/ORADATA/u06/index01.dbf	YES
UNDO02	/home/dba01/ORADATA/u01/UNDO2.dbf	NO

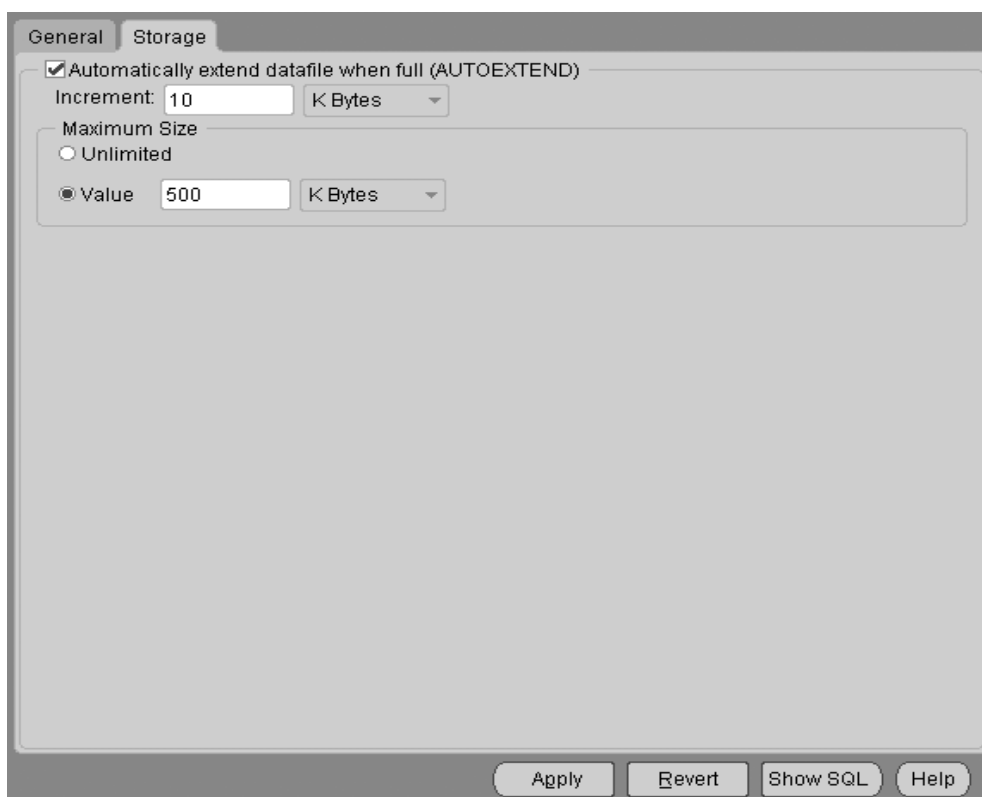
8 rows selected.

Enabling Automatic Extension of Data Files (continued)

Using Oracle Enterprise Manager to Enable Automatic Resizing

From the OEM Console:

1. Navigate to Storage > Datafiles.
2. Select the data file.
3. Select the “Automatically extend datafile when full” check box in the Storage tabbed page.
4. Set the values for the Increment and the Maximum Size.
5. Click Apply.



Manually Resizing a Data File

- **Manually increase or decrease a data file size using ALTER DATABASE.**
- **Resizing a data file adds more space without adding more data files.**
- **Manual resizing of a data file reclaims unused space in database.**
- **Example:**

```
ALTER DATABASE  
DATAFILE '/u03/oradata/userdata02.dbf'  
RESIZE 200M;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Manually Resizing a Data File

Instead of adding space to the database by adding data files, the database administrator can change the size of a data file. Use the ALTER DATABASE command to manually increase or decrease the size of a data file:

```
ALTER DATABASE [database]  
DATAFILE 'filename' [, 'filename']...  
RESIZE integer[K|M]
```

where:

Integer: Is the absolute size, in bytes, of the resulting data file

If there are database objects stored above the specified size, then the data file size is decreased only to the last block of the last objects in the data file.

Adding Data Files to a Tablespace

- Increases the space allocated to a tablespace by adding additional data files
- **ADD DATAFILE** clause is used to add a data file
- **Example:**

```
ALTER TABLESPACE user_data  
ADD DATAFILE '/u01/oradata/userdata03.dbf'  
SIZE 200M;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Adding Data Files to a Tablespace

You can add data files to a tablespace to increase the total amount of disk space allocated for the tablespace with the **ALTER TABLESPACE ADD DATAFILE** command:

```
ALTER TABLESPACE tablespace  
ADD DATAFILE  
filespec [autoextend_clause]
```

Adding Data Files to a Tablespace (continued)

Using Oracle Enterprise Manager Add a Data File

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select the tablespace.
3. Select Add Datafile from the right mouse menu.
4. Enter the file information in the General tabbed page.
5. Click Create.

The screenshot shows a dialog box with two tabs: 'General' and 'Storage'. The 'General' tab is active. It contains the following fields and controls:

- Name:** A text field containing the path `/home/dba01/ORADATA/u01/data03.dbf`.
- Tablespace:** A dropdown menu currently showing `DATA01`.
- Size:** A section containing:
 - File Size:** A text field with the value `200`.
 - Unit:** A dropdown menu showing `M Bytes`.
- Reuse Existing File:** An unchecked checkbox.

At the bottom of the dialog are four buttons: `Create`, `Cancel`, `Show SQL`, and `Help`.

Methods for Moving Data Files

- **ALTER TABLESPACE**
 - Tablespace must be offline.
 - Target data files must exist.

```
ALTER TABLESPACE userdata RENAME  
DATAFILE '/u01/oradata/userdata01.dbf'  
TO '/u02/oradata/userdata01.dbf';
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Methods for Moving Data Files

Depending on the type of tablespace, the database administrator can move data files using one of the following two methods:

The ALTER TABLESPACE Command

The following ALTER TABLESPACE command is applied only to data files in a non-SYSTEM tablespace that does not contain active undo or temporary segments:

```
ALTER TABLESPACE tablespace  
    RENAME DATAFILE 'filename'[, 'filename']...  
    TO 'filename'[, 'filename']...
```

The source filenames must match the names stored in the control file.

Steps to rename a data file:

1. Take the tablespace offline.
2. Use an OS command to move or copy the files.
3. Execute the ALTER TABLESPACE RENAME DATAFILE command.
4. Bring the tablespace online.
5. Use an OS command to delete the file if necessary.

Methods for Moving Data Files

- **ALTER DATABASE**
 - Database must be mounted.
 - Target data file must exist.

```
ALTER DATABASE RENAME  
FILE '/u01/oradata/system01.dbf'  
TO '/u03/oradata/system01.dbf';
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Methods for Moving Data Files (continued)

The ALTER DATABASE Command

The ALTER DATABASE command can be used to move any type of data file:

```
ALTER DATABASE [database]  
  RENAME FILE 'filename'[, 'filename']...  
  TO 'filename'[, 'filename']...
```

Because the SYSTEM tablespace cannot be taken offline, you must use this method to move data files in the SYSTEM tablespace.

Use the following process to rename files in tablespaces that cannot be taken offline:

1. Shut down the database.
2. Use an operating system command to move the files.
3. Mount the database.
4. Execute the ALTER DATABASE RENAME FILE command.
5. Open the database.

Methods for Moving Data Files (continued)

Using Oracle Enterprise Manager to Move a Data File

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select the tablespace in which the data file to be moved resides.
3. Select Offline from the General tabbed page.
4. Click Apply.
5. After the tablespace is offline, update the File Directory information in the General tabbed page.
6. Click Apply.

Note

- These commands verify that the file exists in the new location; they do not create or move files.
- Always provide complete filenames (including their paths) to identify the old and new data files.

Dropping Tablespaces

- **You cannot drop a tablespace if it:**
 - Is the **SYSTEM** tablespace
 - Has active segments
- **INCLUDING CONTENTS** drops the segments.
- **INCLUDING CONTENTS AND DATAFILES** deletes data files.
- **CASCADE CONSTRAINTS** drops all referential integrity constraints.

```
DROP TABLESPACE userdata
INCLUDING CONTENTS AND DATAFILES;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dropping Tablespaces

You can remove a tablespace from the database when the tablespace and its contents are no longer required with the following **DROP TABLESPACE** SQL command:

```
DROP TABLESPACE tablespace
[INCLUDING CONTENTS [AND DATAFILES] [CASCADE CONSTRAINTS]]
```

where:

tablespace: Specifies the name of the tablespace to be dropped

INCLUDING CONTENTS: Drops all the segments in the tablespace

AND DATAFILES: Deletes the associated operating system files

CASCADE CONSTRAINTS: Drops referential integrity constraints from tables outside the tablespace that refer to primary and unique keys in the tables in the dropped tablespace

Dropping Tablespaces (continued)

Guidelines

- A tablespace that still contains data cannot be dropped without the `INCLUDING CONTENTS` option. This option may generate a lot of undo when the tablespace contains many objects.
- After a tablespace has been dropped, its data is no longer in the database.
- When a tablespace is dropped, only the file pointers in the control file of the associated database are dropped. The operating system files still exist and must be deleted explicitly using the appropriate operating system command unless the `AND DATAFILES` clause is used or data files are OMF.
- Even if a tablespace is switched to read-only, it can still be dropped, along with segments within it.
- It is recommended that you take the tablespace offline before dropping it to ensure that no transactions access any of the segments in the tablespace.

Dropping Tablespaces (continued)

Using Oracle Enterprise Manager to Drop a Tablespace

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select the tablespace.
3. Select Remove from the right-mouse menu.
4. Select the “Delete associated datafiles from the OS” check box if you want to delete the data files.

Note: This option only appears with Oracle9i Database Release 2.

5. Click Yes to confirm drop.



Managing Tablespaces Using OMF

- **Define the DB_CREATE_FILE_DEST parameter in one of the following ways:**
 - Initialization parameter file
 - Set dynamically using ALTER SYSTEM command

```
ALTER SYSTEM SET  
db_create_file_dest = '/u01/oradata/dba01';
```

- **When creating the tablespace:**
 - Data file is automatically created and located in DB_CREATE_FILE_DEST
 - Default size is 100 MB
 - AUTOEXTEND is set to UNLIMITED

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Tablespaces Using OMF

When configuring OMF for creating tablespaces, a single initialization parameter, DB_CREATE_FILE_DEST, is specified. The DATAFILE clause is not required. All data files are created automatically and their location is defined by DB_CREATE_FILE_DEST. The data file filename is automatically generated by the Oracle server (ora_tbs1_2ixfh90q.dbf).

Managing Tablespaces Using OMF

- **Creating an OMF tablespace:**

```
CREATE TABLESPACE text_data DATAFILE SIZE 20M;
```

- **Adding an OMF data file to an existing tablespace:**

```
ALTER TABLESPACE text_data ADD DATAFILE;
```

- **Dynamically changing default file location:**

```
ALTER SYSTEM SET  
db_create_file_dest = '/u01/oradata/dba01';
```

- **Dropping a tablespace includes deleting OS files:**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Tablespaces with OMF

Creating OMF Tablespaces

Creating tablespaces with OMF does not require a DATAFILE clause. Tablespaces omitting the DATAFILE clause take the defaults of a 100 MB data file set to AUTOEXTEND with an unlimited MAXSIZE. Optionally a file size can be specified.

```
CREATE TABLESPACE tablespace  
[ DATAFILE [ filename ] [ SIZE integer [K|M] ] ];
```

Adding Data Files to OMF Tablespaces

A data file can be added to an existing tablespace. The ADD DATAFILE command no longer requires file specification.

Dynamically Changing Default File Locations

DB_CREATE_ONLINE_LOG_DEST_n should be set to prevent log files and control files from being placed with data files. The destination can be changed dynamically using the ALTER SYSTEM SET command.

Dropping OMF Tablespaces

Data files from OMF created tablespaces are deleted at the OS level when the associated tablespace is dropped.

Obtaining Tablespace Information

Obtaining tablespace and data file information can be obtained by querying the following:

- **Tablespace information:**

- DBA_TABLESPACES
- V\$TABLESPACE

- **Data file information:**

- DBA_DATA_FILES
- V\$DATAFILE

- **Temp file information:**

- DBA_TEMP_FILES
- V\$TEMPFILE

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Summary

In this lesson, you should have learned how to:

- **Use tablespaces to separate data**
- **Create various types of tablespaces**
- **Manage tablespaces**
- **Manage tablespaces using OMF**
- **Obtain tablespace information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 8 Overview

This practice covers the following topics:

- **Creating tablespaces**
- **Modifying tablespaces**
- **Configuring for and creating a tablespace using OMF**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 8 Overview

Note: This practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 8: Managing Tablespaces and Data Files

1 Create permanent tablespaces with the following names and storage:

a Tablespace name: DATA01

Data file name: data01.dbf

Size: 2M

Extent Management: dictionary

Location: u04

b Tablespace name: DATA02

Data file name: data02.dbf

Size: 1M

Extent management: local uniform size 100K

Location: u03

c Tablespace name: INDEX01

Data file name: index01.dbf

Size: 1M

Extent management: local uniform size 4K

Location: u02

Enable automatic extension of 500 KB when more extents are required with a maximum size of 2 MB.

d Tablespace name: RONLY

Data file name: ronly01.dbf

Size: 1M

Location: u01

Default storage. DO NOT make the tablespace read-only at this time.

e Display the information from the data dictionary.

Hint: Information about tablespaces can be viewed using any of the following queries.

- DBA_TABLESPACES
- V\$TABLESPACE
- V\$DATAFILE

Practice 8: Solutions (continued)

- 2 Allocate 500K more disk space to tablespace DATA02. Verify the result.
- 3 Relocate tablespace INDEX01 to subdirectory u06. Verify relocation and status of INDEX01.

Hints

- Take the INDEX01 tablespace offline.
 - Use V\$DATAFILE to verify status.
 - Use operating system move command to move the tablespace to u06.
 - Use ALTER TABLESPACE to relocate the tablespace.
 - Place the INDEX01 tablespace online.
 - Use V\$DATAFILE to verify status.
- 4 a Connect as user SYSTEM and create a table in tablespace RONLY. Make tablespace read-only. Run a query to verify it.
 - b Attempt to create an additional table called TABLE2. Drop the first created table, TABLE1. What happens?
- 5 Drop tablespace RONLY and the associated data file. Verify it.

Hints

- Use the INCLUDING CONTENTS AND DATAFILES clause
 - View V\$TABLESPACE to verify the tablespace was dropped.
 - Use the ! from the SQL prompt to see a list of the data files in u01.
- 6 Set DB_CREATE_FILE_DEST to \$HOME/ORADATA/u05 in memory only. Create tablespace DATA03 size 5M. Do not specify a file location. Verify the creation of the data file.

9

Storage Structure and Relationships

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Describe the logical structure of the database**
- **List the segment types and their uses**
- **List the keywords that control block space usage**
- **Obtain storage structure information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

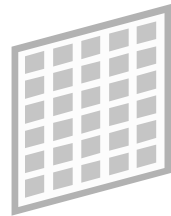
Storage and Relationship Structure

Database																			
PROD																			
TABLESPACES SYSTEM					USER_DATA								RBS				TEMP		
DATA FILES DISK1/SYS1.dbf					DISK2/ USER1.dbf				DISK3/ USER2.dbf				DISK1/ UNDO1.dbf				DISK1/ TEMP.dbf		
SEGMENTS D.D. Table		D.D. Index		RB Seg	S_DEPT Data Seg	S_EMP Data Seg	S_DEPT (cont'd) Data Seg	S_EMP FIRST_N AME Index Index Seg		RBS1 RB Seg	RBS2 RB Seg	RBS1 (cont'd) RB Seg	RBS2 (cont'd) RB Seg	Temp Temp Seg					
Data Seg	Index Seg																		
EXTENTS																			
1	2	1	2	1	2	1	1	2	2	1	FREE	1	1	2	2	1			
Oracle DATA BLOCKS																			

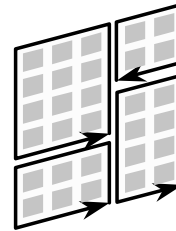
ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Types of Segments



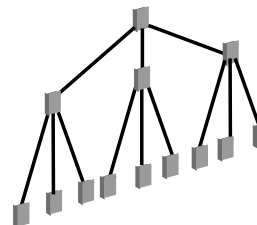
Table



**Table
partition**



Cluster



Index

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Types of Segments

Segments are space-occupying objects in a database. They use space in the data files of a database. This section describes the different types of segments.

Table

A table is the most common means of storing data within a database. A table segment stores that data for a table that is neither clustered nor partitioned. Data within a table segment is stored in no particular order, and the database administrator (DBA) has very little control over the location of rows within the blocks in a table. All the data in a table segment must be stored in one tablespace.

Table partition

Scalability and availability are major concerns when there is a table in a database with high concurrent usage. In such cases, data within a table may be stored in several partitions, each of which resides in a different tablespace. The Oracle server currently supports partitioning by a range of key values, by a hashing algorithm, and by a list of values. If a table is partitioned, each partition is a segment, and the storage parameters can be specified to control them independently. Use of this type of segment requires the partitioning option within the Oracle9i Enterprise Edition.

Types of Segments (continued)

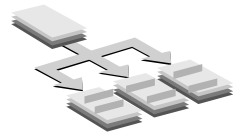
Cluster

A cluster, like a table, is a type of data segment. Rows in a cluster are stored based on key column values. A cluster may contain one or more tables. Tables in a cluster belong to the same segment and share the same storage characteristics. The rows in a clustered table can be accessed with an index or hashing algorithm.

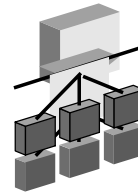
Index

All the entries for a particular index are stored within one index segment. If a table has three indexes, three index segments are used. The purpose of this segment is to look up the location of rows in a table based on a specified key.

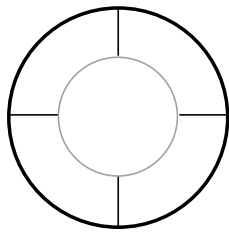
Types of Segments



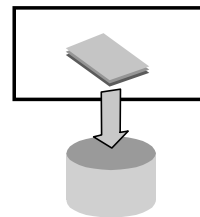
**Index-organized
table**



**Index
partition**



**Undo
segment**



**Temporary
segment**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Types of Segments (continued)

Index-Organized Table

In an index-organized table, data is stored within the index based on the key value. An index-organized table does not need a table lookup, because all the data can be retrieved directly from the index tree.

Index Partition

An index can be partitioned and spread across several tablespaces. In this case, each partition in the index corresponds to a segment and cannot span multiple tablespaces. The primary use of a partitioned index is to minimize contention by spreading index input/output (I/O). Use of this type of segment requires the partitioning option within the Oracle9i Enterprise Edition.

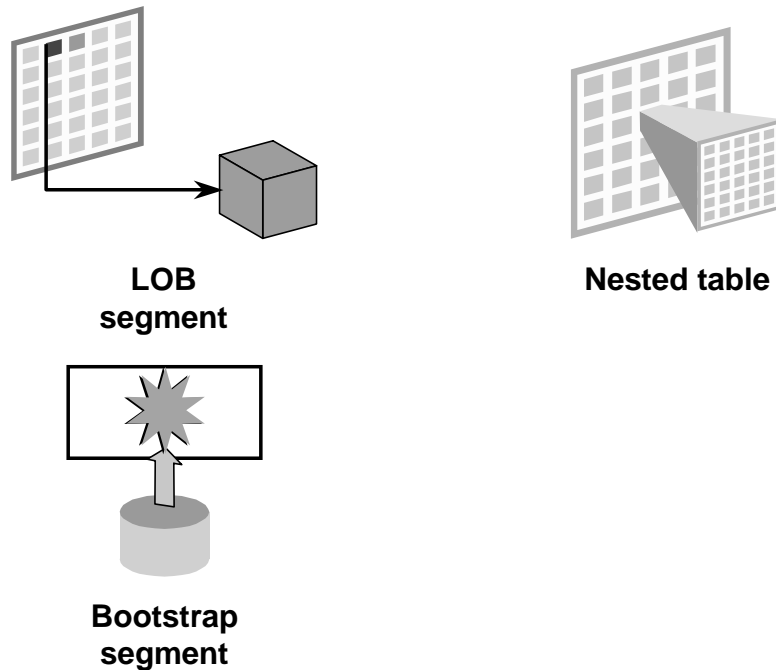
Undo Segment

An undo segment is used by a transaction that is making changes to a database. Before changing the data or index blocks, the old value is stored in the undo segment. This allows a user to undo changes made.

Temporary Segment

When a user executes commands such as `CREATE INDEX`, `SELECT DISTINCT`, and `SELECT GROUP BY`, the Oracle server tries to perform sorts in memory. When a sort needs more space than the space available in memory, intermediate results are written to the disk. Temporary segments are used to store these intermediate results.

Types of Segments



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Types of Segments (continued)

LOB Segment

One or more columns in a table can be used to store large objects (LOBs) such as text documents, images, or videos. If the column is large, the Oracle server stores these values in separate segments known as LOB segments. The table contains only a locator or a pointer to the location of the corresponding LOB data.

Nested Table

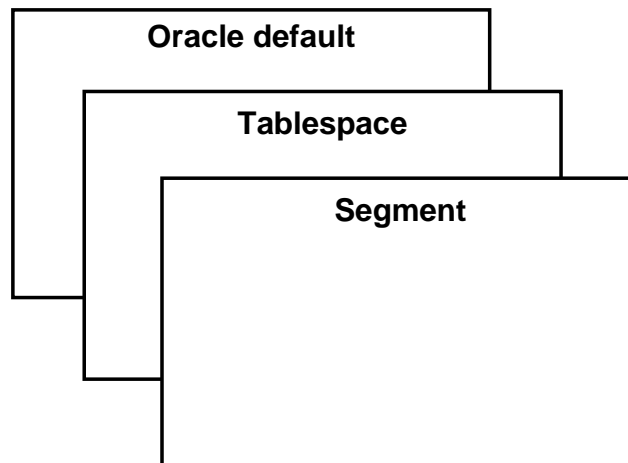
A column in a table may be made up of a user-defined table as in the case of items within an order. In such cases, the inner table, which is known as a nested table, is stored as a separate segment.

Bootstrap Segment

A bootstrap segment, also known as a cache segment, is created by the `sql .bsq` script when a database is created. This segment helps to initialize the data dictionary cache when the database is opened by an instance.

The bootstrap segment cannot be queried or updated and does not require any maintenance by the database administrator.

Storage Clause Precedence



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Storage Clause Parameters

A storage clause can be specified at the segment level to control how extents are allocated to a segment.

- Any storage parameter specified at the segment level overrides the corresponding option set at the tablespace level, except for the `MINIMUM EXTENT` or `UNIFORM SIZE` tablespace parameter.
- When storage parameters are not set explicitly at the segment level, they default to those at the tablespace level.
- When storage parameters are not set explicitly at the tablespace level, the Oracle server system defaults are used.

Other Considerations

- If storage parameters are altered, the new options apply only to the extents not yet allocated.
- Some parameters cannot be specified at the tablespace level. These parameters must be specified at the segment level only.
- If minimum extent size has been specified for the tablespace, this size applies to all extents that are allocated for segments in the tablespace in the future.

Extent Allocation and Deallocation

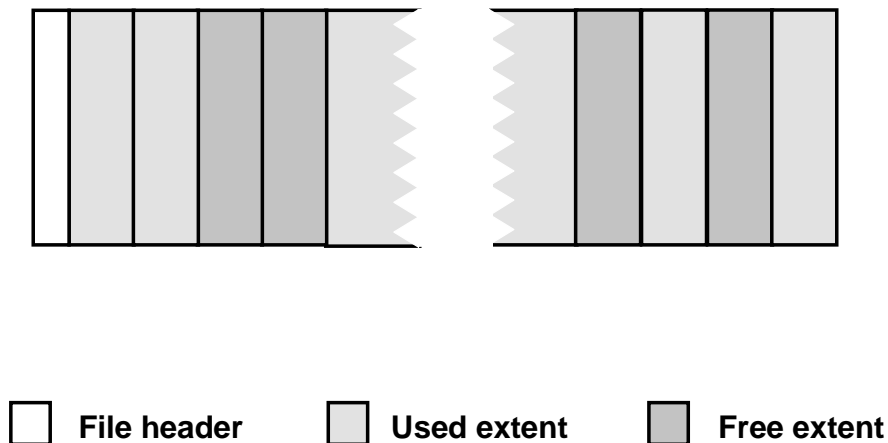
- **An extent is a chunk of space used by a segment within a tablespace.**
- **An extent is allocated when the segment is:**
 - **Created**
 - **Extended**
 - **Altered**
- **An extent is deallocated when the segment is:**
 - **Dropped**
 - **Altered**
 - **Truncated**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Used and Free Extents

Data file



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Used and Free Extents

When a tablespace is created, the data files in the tablespace contains a header, which is the first block or blocks in the file.

As segments are created, they are allocated space from the free extents in a tablespace. Contiguous space used by a segment is referred to as a used extent. When segments release space, the extents that are released are added to the pool of free extents available in the tablespace.

Database Block

- **Minimum unit of I/O**
- **Consists of one or more operating system blocks**
- **Set at tablespace creation**
- **DB_BLOCK_SIZE is the default block size**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Multiple Block Size Support

- A database can be created with a standard block size and up to four nonstandard block sizes.
- Block sizes can have any power-of-two value between 2 KB and 32 KB.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Multiple Block Size Support

Oracle9i supports the creation of databases with multiple block sizes. This feature is useful in the following situations:

- When transporting a tablespace from an online transactional processing (OLTP) database to an enterprise data warehouse. Oracle9i facilitates transport between databases of different block sizes
- When you need to locate objects in tablespaces of appropriate block sizes in order to maximize I/O performance

The block size of the SYSTEM tablespace is termed the standard block size. This is set when the database is created. With Oracle9i you can specify up to four nonstandard block sizes, in addition to a standard block size. In the initialization file, you can configure subcaches within the buffer cache for each of these block sizes. Subcaches can also be configured while an instance is running. You can create tablespaces having any of these block sizes. The standard block size is used for the system tablespace and most other tablespaces.

Standard Block Size

- **Set at database creation using the `DB_BLOCK_SIZE` parameter; cannot be changed without re-creating the database**
- **Used for `SYSTEM` and `TEMPORARY` tablespaces**
- **`DB_CACHE_SIZE` specifies the size of the `DEFAULT` buffer cache for standard block size:**
 - **Minimum size = one granule (4 MB or 16 MB)**
 - **Default value = 48 MB**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Standard Block Size

The `DB_BLOCK_SIZE` initialization parameter specifies the standard block size for the database. This block size is used for the `SYSTEM` tablespace, and for any temporary tablespace. Unless specified otherwise, the standard block size is also used as the default block size for a tablespace. Oracle can support up to four additional nonstandard block sizes.

The most-commonly used block size should be picked as the standard block size. In many cases, this is the only block size that you need to specify. Typically, `DB_BLOCK_SIZE` is set to either 4 KB or 8 KB. If not specified, the default data block size is operating system specific and is generally adequate.

The block size cannot be changed after database creation, except by re-creating the database.

The `DB_CACHE_SIZE` initialization parameter replaces the `DB_BLOCK_BUFFERS` initialization parameter that was used in previous releases. The `DB_CACHE_SIZE` parameter specifies the size of the cache of standard block size buffers, where the standard block size is specified by `DB_BLOCK_SIZE`.

Standard Block Size (continued)

For backward compatibility the `DB_BLOCK_BUFFERS` parameter still works, but it remains a static parameter and cannot be combined with any of the dynamic sizing parameters.

Note: A granule is a unit of contiguous virtual memory allocation. The size of a granule depends on the estimated total SGA size whose calculation is based on the value of the `SGA_MAX_SIZE` parameter: 4 MB if estimated SGA size is < 128 MB, 16 MB otherwise.

Nonstandard Block Size

- **Configure additional caches with the following dynamic parameters:**
 - **DB_2K_CACHE_SIZE** for 2 KB blocks
 - **DB_4K_CACHE_SIZE** for 4 KB blocks
 - **DB_8K_CACHE_SIZE** for 8 KB blocks
 - **DB_16K_CACHE_SIZE** for 16 KB blocks
 - **DB_32K_CACHE_SIZE** for 32 KB blocks
- **DB_nK_CACHE_SIZE is not allowed if nK is the standard block size.**
- **Minimum size for each cache is one granule.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Nonstandard Block Size

The Database Buffer Cache initialization parameters determine the size of the database buffer cache component of SGA. You use them to specify the sizes of caches for the various block sizes used by the database. If you intend to use multiple block sizes in your database, you must have the `DB_CACHE_SIZE` and at least one `DB_nK_CACHE_SIZE` parameter set. Each parameter specifies the size of the buffer cache for the corresponding block size. The default value for `DB_nK_CACHE_SIZE` parameter is zero. Do not set this parameter to zero if there are any online tablespaces with an n KB block size.

Platform-specific block size restrictions apply. For example, you cannot set `DB_32K_CACHE_SIZE` if the maximum block size on the platform is less than 32 KB. Also, you cannot set `DB_2K_CACHE_SIZE` if the minimum block size is greater than 2 KB.

Note: These parameters cannot be used to size the cache for the standard block size. For example, if the value of `DB_BLOCK_SIZE` is 2 KB, it is illegal to set `DB_2K_CACHE_SIZE`. The size of the cache for the standard block size is always determined from the value of `DB_CACHE_SIZE`.

Nonstandard Block Size (continued)

Using Oracle Enterprise Manager to Configure Additional Caches

From the OEM Console:

1. Navigate to Configuration.
2. Select All Initialization Parameters.
3. Make appropriate changes to the following parameters:
 - DB_2K_CACHE_SIZE
 - DB_4K_CACHE_SIZE
 - DB_8K_CACHE_SIZE
 - DB_16K_CACHE_SIZE
 - DB_32K_CACHE_SIZE
4. Click OK.

Creating Nonstandard Block Size Tablespaces

```
CREATE TABLESPACE tbs_1
DATAFILE 'tbs_1.dbf'
SIZE 10M BLOCKSIZE 4K;
```

```
DESCRIBE dba_tablespaces
Name Null? Type
-----
TABLESPACE_NAME NOT NULL VARCHAR2(30)
BLOCK_SIZE NOT NULL NUMBER
...
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating Nonstandard Block Size Tablespaces

Use the `BLOCKSIZE` clause to specify a nonstandard block size for the tablespace. You can specify the size in bytes or in kilobytes, using the `K` suffix.

In order to specify this clause, you must have the `DB_CACHE_SIZE` and at least one `DB_nK_CACHE_SIZE` parameter set, and the integer that you specify in this clause must correspond with the setting of one `DB_nK_CACHE_SIZE` parameter setting.

Restriction: You cannot specify nonstandard block sizes for a temporary tablespace (that is, if you also specify `TEMPORARY`) or if you intend to assign this tablespace as the temporary tablespace for any users.

The first statement above creates a new tablespace called `tbs_1` with the file `tbs_1.dbf` having a block size of 4 KB. In order for this statement to succeed, the buffers of size 4 KB must currently be configured in the buffer cache.

Note: A new column has been added to the `*_TABLESPACES` dictionary views in order to reflect the corresponding block size used in a particular tablespace.

Creating Nonstandard Block Size Tablespaces (continued)

Using Oracle Enterprise Manager to Create a Nonstandard Block Size Tablespace

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select Create from the right-mouse menu.
3. Enter information to create the tablespace
4. Select the Storage page and state the value for Block Size.
5. Click Create.

The screenshot shows the 'Storage' tab of the Oracle Enterprise Manager console. It contains the following elements:

- General** and **Storage** tabs at the top.
- Extent Management:** Two radio buttons: ☒ Locally managed and ☐ Managed in the dictionary.
- Allocation:** Two radio buttons: ☒ Automatic Allocation and ☐ Uniform Allocation.
- Size:** A text input field and a dropdown menu currently set to 'K Bytes'.
- Enable logging:** Two radio buttons: ☒ Yes - Generates redo logs and recoverable, and ☐ No - Faster updates, no redo logs generated and not recoverable.
- Block Size:** A dropdown menu set to '4096' and a label 'Bytes'.
- Buttons:** 'Create', 'Cancel', 'Show SQL', and 'Help' at the bottom.

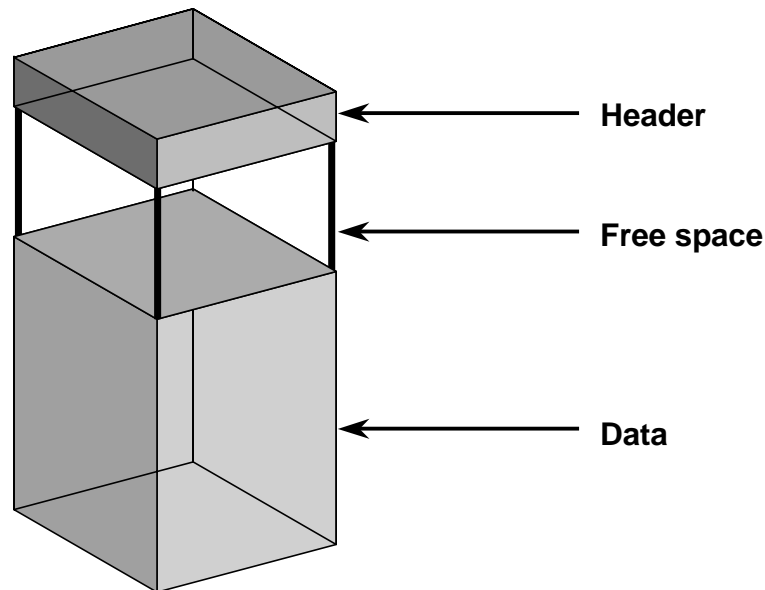
Multiple Block Sizing Rules

- All partitions of a partitioned object must reside in tablespaces of the same block size.
- All temporary tablespaces, including the permanent ones that are being used as default temporary tablespaces, must be of standard block size.
- Index-organized table overflow and out-of-line LOB segments can be stored in a tablespace with a block size different from the base table.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Database Block Contents



ORACLE

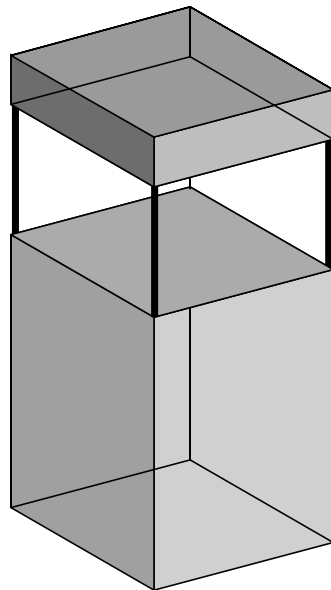
Copyright © Oracle Corporation, 2002. All rights reserved.

Database Block Contents

Oracle data blocks contain:

- **Block header:** The header contains the data block address, table directory, row directory, and transaction slots that are used when transactions make changes to rows in the block. Block headers grow from the top down.
- **Data space:** Row data is inserted into the block from the bottom up.
- **Free space:** The free space in a block is in the middle of the block. Thus both the header and the row data space can grow when necessary. The free space in a block is contiguous initially. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the Oracle server when necessary.

Block Space Utilization Parameters



INITRANS

MAXTRANS

PCTFREE

PCTUSED

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Block Space Utilization Parameters

Block space utilization parameters can be used to control the use of space in data and index segments.

Parameters Controlling Concurrency

INITRANS and MAXTRANS: Specify the initial and the maximum number of transaction slots that are created in an index or a data block. The transaction slots are used to store information about transactions that are making changes to the block at a point in time. A transaction uses only one transaction slot, even if it is changing more than one row or index entry.

INITRANS: Guarantees a minimum level of concurrency. It defaults to 1 for a data segment and 2 for an index segment, guarantees a minimum level of concurrency. For example, if set to 3, INITRANS ensures that at least three transactions can concurrently make changes to the block. If necessary, additional transaction slots can be allocated from the free space in the block to permit more concurrent transactions to modify rows in the block.

MAXTRANS: Default value is 255, sets the limit for the number of concurrent transactions that can make changes to a data or an index block. When set, this value restricts use of space for transaction slots and therefore guarantees that there is sufficient space in the block for use by row or index data.

Block Space Utilization Parameters (continued)

Parameters Controlling the Use of Data Space

PCTFREE: This parameter specifies for a data segment, the percentage of space in each data block that is reserved for growth resulting from updates to rows in the block. The default for PCTFREE is 10%.

PCTUSED: For a data segment, this parameter represents the minimum percentage of used space that the Oracle server tries to maintain for each data block of the table. A block is put back on the free list when its used space falls below PCTUSED. The free list of a segment is a list of blocks that are candidates for accommodating future inserts. A segment, by default, is created with one free list. Segments can be created with a higher number of free lists by setting the FREELISTS parameter of the storage clause. The default for PCTUSED is 40%.

Both PCTFREE and PCTUSED are calculated as percentages of available data space, that is, the block space that remains after deducting the header space from the total block size.

Note: The use of these parameters for indexes is discussed in detail in the “Managing Indexes” lesson.

Specifying FREELISTS is discussed in detail in the course *Oracle9i: Performance Tuning*.

Data Block Management

Two methods are available for managing data blocks:

- **Automatic segment-space management**
- **Manual management**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Segment-Space Management

- It is a method of managing free space inside database segments.
- Tracking in-segment free and used space is done using bitmaps as opposed to using free lists.
- This method provides:
 - Ease of management
 - Better space utilization
 - Better performance for concurrent INSERT operations

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Segment-Space Management

Ease of Use

PCTUSED, FREELISTS, FREELIST GROUPS are managed automatically.

Better Space Utilization

All objects, and especially objects with greatly varying row sizes, utilize space more efficiently.

Better Concurrency Handling

Run-time adjustments to variations in concurrent access are improved.

Note: Prior to Oracle9i Database Release 2, LOBs were not automatic segment-space managed.

Automatic Segment-Space Management

- **Bitmap segments contain a bitmap that describes the status of each block in the segment with respect to its available space.**
- **The map is contained in a separate set of blocks referred to as bitmapped blocks (BMBs).**
- **When inserting a new row, the server searches the map for a block with sufficient space.**
- **As the amount of space available in a block changes, its new state is reflected in the bitmap.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Configuring Automatic Segment-Space Management

- Automatic segment-space management can be enabled at the tablespace level only, for locally managed tablespaces.

```
CREATE TABLESPACE data02
DATAFILE '/u01/oradata/data02.dbf' SIZE 5M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K
SEGMENT SPACE MANAGEMENT AUTO;
```

- After a tablespace is created, the specifications apply to all segments created in the tablespace.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Configuring Automatic Segment-Space Management

Bitmapped segments are specified through the `SEGMENT SPACE MANAGEMENT AUTO` clause of the `CREATE TABLESPACE` command, which cannot be subsequently altered. Any specifications of `PCTUSED`, `FREELIST`, and `FREELIST GROUPS` are ignored if defined.

Segments that can be bitmap-managed are regular tables, indexes, index-organized tables (IOT), and LOBs.

Note: Prior to Oracle9i Database Release 2, LOBs were not Automatic Segment-Space managed.

Manual Data Block Management

- **Allows you to configure data blocks manually using parameters such as:**
 - **PCTFREE**
 - **PCTUSED**
 - **FREELIST**
- **The only method available in previous Oracle versions**

ORACLE

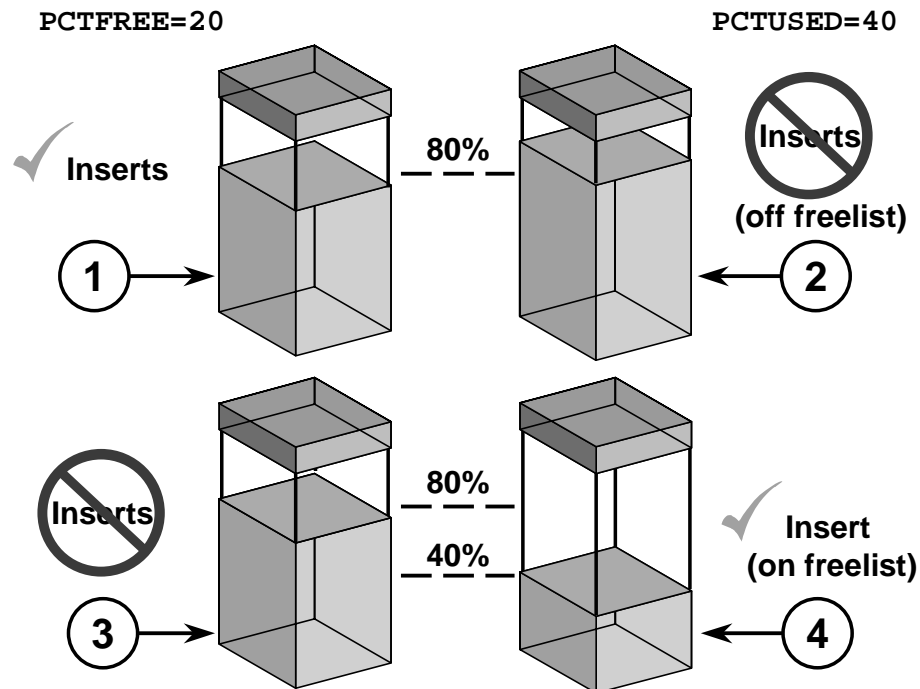
Copyright © Oracle Corporation, 2002. All rights reserved.

Manual Data Block Management

With manual database block management you can configure how block space is used and when a block is available. Parameters such as **PCTFREE**, **PCTUSED**, and **FREELIST** are used in manual management. Previously this was the only method available for managing data blocks.

The manual method is the default.

Block Space Usage



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Block Space Usage

The following steps explain how space within a block is managed for a data segment with PCTFREE=20 and PCTUSED=40:

1. Rows are inserted into the block until the free space in the block is equal to or less than 20%. The block is no longer available for inserts when rows occupy 80% (100 – PCTFREE) or more of the available data space in the block.
2. The remaining 20% can be used when the size of a row increases. For example, a column that was originally NULL is updated to be assigned a value. Thus block utilization may be in excess of 80% as a result of updates.
3. If rows are deleted in the block or if rows decrease in size as a result of updates, block utilization may fall below 80%. However, a block is not used for inserts until the utilization falls below PCTUSED, which in this example, is 40%.
4. When the utilization falls below PCTUSED, the block is available for inserts. As rows are inserted into the block, the utilization of the block increases and the cycle repeats starting with step 1.

Note: Guidelines for setting PCTFREE and PCTUSED are discussed in the lessons on tables and indexes, “Managing Tables” and “Managing Indexes,” respectively.

Obtaining Storage Information

Information about storage can be obtained by querying the following views:

- **DBA_EXTENTS**
- **DBA_SEGMENTS**
- **DBA_TABLESPACES**
- **DBA_DATA_FILES**
- **DBA_FREE_SPACE**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Querying the Data Dictionary

The relationships between tablespaces, data files, segments, and free and used extents can be viewed by querying the data dictionary.

When a tablespace with one or more files is created, a row is added to **DBA_TABLESPACES**. For each file in the database, a row is added to **DBA_DATA_FILES**. At this stage, the space in each data file, excluding the file header, shows up as one free extent in **DBA_FREE_SPACE**.

When a segment is created, a row is visible in **DBA_SEGMENTS**. When a segment is created, a row is visible in **DBA_SEGMENTS**. The space allocated to the extents in this segment can be viewed from **DBA_EXTENTS**. **DBA_FREE_SPACE** is adjusted to show lower free space in the files where the extents have been created for the segment.

All the space in a file (excluding the header block) must be accounted for either in **DBA_FREE_SPACE** or in **DBA_EXTENTS**.

Querying the Data Dictionary (continued)

DBA_SEGMENTS View

Query the DBA_SEGMENTS view to get the number of extents and blocks allocated to a segment.

```
SQL> SELECT segment_name, tablespace_name, extents, blocks
2 FROM dba_segments
3 WHERE owner = 'HR';
```

SEGMENT_NAME	TABLESPACE	EXTENTS	BLOCKS
REGIONS	SAMPLE	1	8
LOCATIONS	SAMPLE	1	8
DEPARTMENTS	SAMPLE	1	8
JOBS	SAMPLE	1	8
EMPLOYEES	SAMPLE	1	8
JOB_HISTORY	SAMPLE	1	8

5 rows selected.

DBA_EXTENTS View

Use the DBA_EXTENTS view to check the extents for a given segment.

```
SQL> SELECT extent_id, file_id, block_id, blocks
2 FROM dba_extents
3 WHERE owner='HR'
4 AND segment_name='EMPLOYEES';
```

EXTENT_ID	FILE_ID	BLOCK_ID	BLOCKS
0	4	2	5
1	4	27	5
2	4	32	10
3	4	42	15
4	4	57	20

5 rows selected.

Querying the Data Dictionary (continued)

DBA_FREE_SPACE View

Use the DBA_FREE_SPACE view to check the extents for a given segment.

```
SQL> SELECT tablespace_name, count(*),  
2> max(blocks), sum(blocks)  
3> FROM dba_free_space  
4> GROUP BY tablespace_name;
```

TABLESPACE_NAME	COUNT(*)	MAX(BLOCKS)	SUM(BLOCKS)
-----	-----	-----	-----
DATA01	2	1284	1533
RBS	3	2329	2419
SORT	1	1023	1023
SYSTEM	1	5626	5626
TEMP	1	2431	2431

5 rows selected.

Summary

In this lesson, you should have learned how to:

- **Use tablespaces to:**
 - Separate segments to ease administration
 - Control the user's space allocation
- **Categorize segments by the type of information stored in the segment**
- **Determine extent sizes using the storage clause**
- **Control block space utilization**
- **Obtain storage structure information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 9 Overview

This practice covers identifying and obtaining information on the various types of storage structures in the database.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 9 Overview

Note: This practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 9: Storage Structure and Relationships

- 1 As user SYSTEM, run the lab09_01.sql script to create tables and indexes.
- 2 Identify the different types of segments in the database.
- 3 Write a query to check which segments are within five extents short of the maximum extents. Ignore the bootstrap segment. This query is useful in identifying any segments that are likely to generate errors during future data load.

Hints

- Select from DBA_EXTENTS.
 - Use the segment_name, segment_type, max_extents, extents keywords.
- 4 Which files have space allocated for the EMP table?
 - 5 Run the lab09_05.sql script.
 - 6 List the free space available by tablespace. The query should display the number of fragments, the total free space, and the largest free extent in each tablespace.
 - 7 List segments that will generate errors because of lack of space when they try to allocate an additional extent.

10

Managing Undo Data

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Describe the purpose of undo data**
- **Implement Automatic Undo Management**
- **Create and configure undo segments**
- **Obtain undo segment information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Undo Data

- There are two methods for managing undo data:
 - Automatic Undo Management
 - Manual Undo Management
- The term *undo* was known as *rollback* in previous versions.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Undo Data

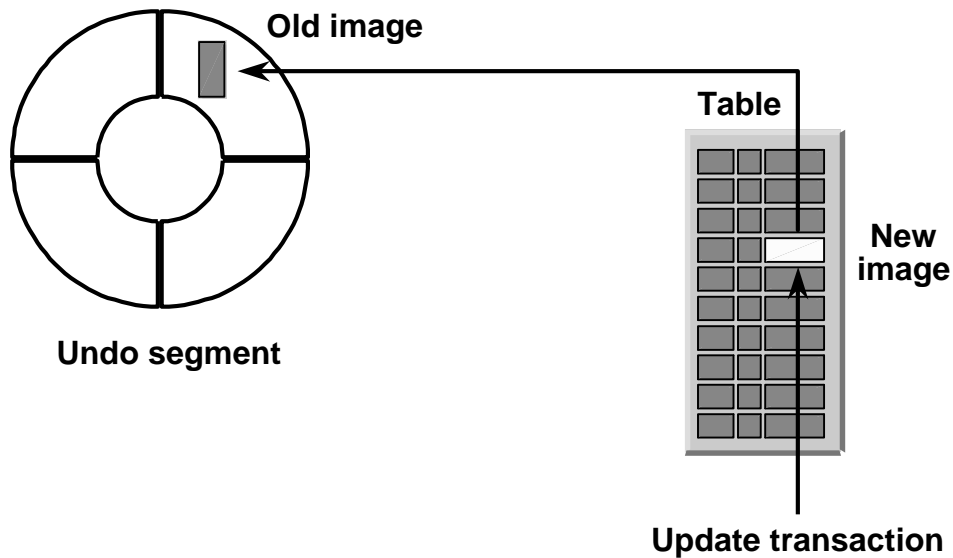
Automatic Undo Management

The Oracle server automatically manages the creation, allocation, and tuning of undo segments.

Manual Undo Management

You manually manage the creation, allocation, and tuning of undo segments. It was the only method available prior to Oracle9i. Information about manual undo management can be found in “Appendix B: Manually Managing Undo Data.”

Undo Segment



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Undo Segment

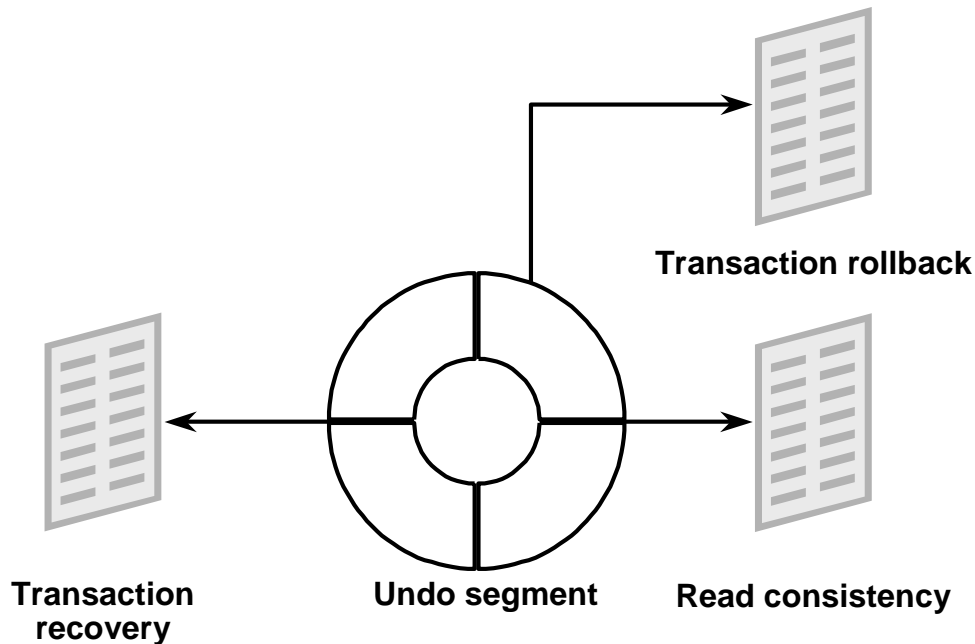
An undo segment is used to save the old value (undo data) when a process changes data in a database. It stores the location of the data and the data as it existed before being modified.

The header of an undo segment contains a transaction table where information about the current transactions using the undo segment is stored.

A serial transaction uses only one undo segment to store all of its undo data.

Many concurrent transactions can write to one undo segment.

Undo Segments: Purpose



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Undo Segments: Purpose

Transaction Rollback

When a transaction modifies a row in a table, the old image of the modified columns (undo data) is saved in the undo segment. If the transaction is rolled back, the Oracle server restores the original values by writing the values in the undo segment back to the row.

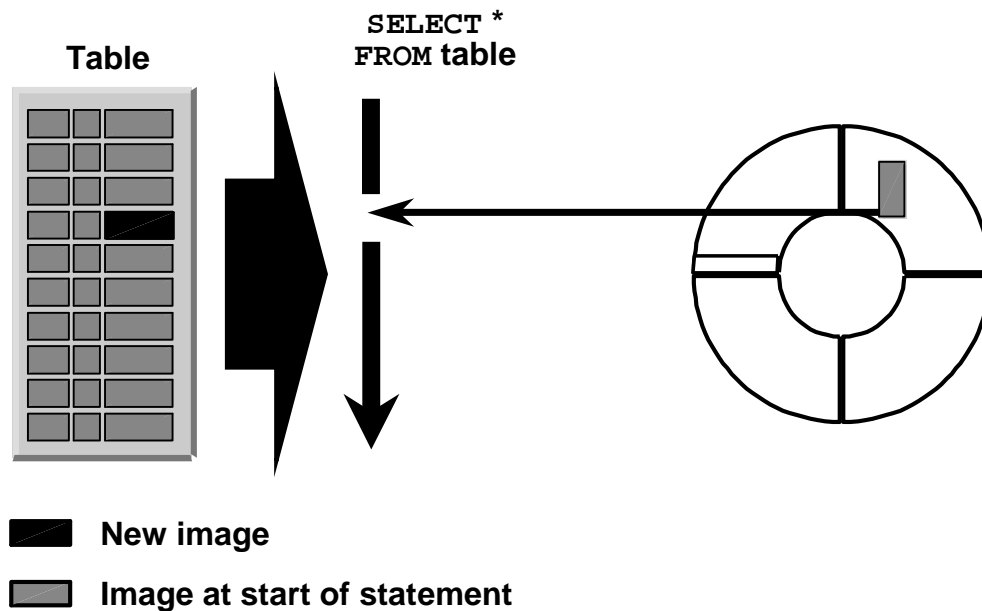
Transaction Recovery

If the instance fails while transactions are in progress, the Oracle server needs to undo any uncommitted changes when the database is opened again. This rollback is part of transaction recovery. Recovery is possible only because changes made to the undo segment are also protected by the online redo log files.

Read Consistency

While transactions are in progress, other users in the database should not see any uncommitted changes made by these transactions. In addition, a statement should not see any changes that were committed after the statement begins execution. The old values (undo data) in the undo segments are also used to provide the readers a consistent image for a given statement.

Read Consistency



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Read Consistency

The Oracle server guarantees that a statement sees data from a consistent time, even if that data is modified by other transactions.

When the Oracle server begins executing a `SELECT` statement, it determines the current system change number (SCN) and ensures that any changes not committed before this SCN are not processed by the statement. Consider the case where a long-running query is executed at a time when several changes are being made. If a row has changes that were not committed at the start of the query, the Oracle server constructs a read-consistent image of the row by retrieving the before image of the changes from the undo segment and applying the changes to a copy of the row in memory.

Transaction Read Consistency

Read consistency is always provided for a SQL statement. However, you can request read consistency for a read-only transaction by issuing the following command at the beginning of the transaction:

```
SQL> SET TRANSACTION READ ONLY;
```

Or, you can request read consistency for a transaction performing DML by issuing the following command at the beginning of the transaction:

```
SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

In either case, the Oracle server provides data that is read consistent from the start of the transaction. Using `SERIALIZABLE` can have a negative impact on performance.

Types of Undo Segments

- **SYSTEM:** Used for objects in the **SYSTEM** tablespace
- **Non-SYSTEM:** Used for objects in other tablespaces:
 - **Auto mode:** Requires an **UNDO** tablespace
 - **Manual mode:**
 - Private:** Acquired by a single instance
 - Public:** Acquired by any instance
- **Deferred:** Used when tablespaces are taken offline immediate, temporary, or for recovery

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Types of Undo Segments

SYSTEM Undo Segment

The **SYSTEM** undo segment is created in the **SYSTEM** tablespace when a database is created. This undo segment can be used only for changes made to objects in the **SYSTEM** tablespace. The **SYSTEM** undo segment exists and works the same in both manual and auto mode.

Non-SYSTEM Undo Segments

A database that has multiple tablespaces needs at least one non-**SYSTEM** undo segment for manual mode or one **UNDO** tablespace for auto mode.

Manual Mode

In manual mode, a non-**SYSTEM** undo segment, which is created by the database administrator, can be used for changes made to objects in any non-**SYSTEM** tablespace. There are two types of non-**SYSTEM** undo segments.

Private

Private undo segments are segments that are brought online by an instance because they are listed in the parameter file. However, they can be brought online explicitly by issuing an **ALTER ROLLBACK SEGMENT** command.

Types of Undo Segments (continued)

Public

Public undo segments form a pool of undo segments available in a database. Public undo segments are normally used with the Oracle Real Application Clusters to create a pool of undo segments that can be used by any of the Real Application Clusters instances.

Note: The use of public undo segments is discussed in the *Oracle9i Real Application Clusters and Administration* manual.

Deferred Undo Segments

Deferred undo segments may be created when a tablespace is brought offline. They are used to roll back transactions when the tablespace is brought back online. They are dropped automatically when they are no longer needed.

Because deferred undo segments are maintained by the Oracle server, no maintenance is required on your part.

Automatic Undo Management: Concepts

- Undo data is managed using an UNDO tablespace.
- You allocate one UNDO tablespace per instance with enough space for the workload of the instance.
- The Oracle server automatically maintains undo data within the UNDO tablespace.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Concepts

Undo segments are created with the naming convention:

`_SYSSMUn$`

For example:

`_SYSSMU1$`

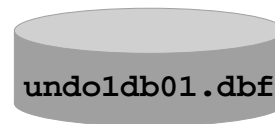
`_SYSSMU2$`

Automatic Undo Management: Configuration

- **Configure two parameters in the initialization file:**
 - UNDO_MANAGEMENT
 - UNDO_TABLESPACE
- **Create at least one UNDO tablespace.**



**Initialization
file**



UNDO tablespace

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Configuration

If only one UNDO tablespace exists in the database and UNDO_MANAGEMENT is set to AUTO, then the UNDO_TABLESPACE parameter is optional; the Oracle server will automatically choose the UNDO tablespace.

Automatic Undo Management: Initialization Parameters

- **UNDO_MANAGEMENT:** Specifies whether the system should use **AUTO** or **MANUAL** mode
- **UNDO_TABLESPACE:** Specifies a particular **UNDO** tablespace to be used

```
UNDO_MANAGEMENT=AUTO  
UNDO_TABLESPACE=UNDOTBS
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Initialization Parameters

UNDO_MANAGEMENT Parameter

The **UNDO_MANAGEMENT** parameter determines the undo mode of the database. The parameter can be set to one of two values, **AUTO** or **MANUAL**, and must be set in the initialization parameter file. **UNDO_MANAGEMENT** cannot be changed dynamically after the database starts. **AUTO** mode sets the database to automatic undo management and requires an **UNDO** tablespace. In **MANUAL** mode, the default value, you can create and manage undo segments when needed within the database as in previous versions of the Oracle server.

UNDO_TABLESPACE Parameter

Specifies the **UNDO** tablespace to be used. This parameter can be set in the initialization files or altered dynamically using the **ALTER SYSTEM** command.

```
SQL> ALTER SYSTEM SET undo_tablespace = UNDOTBS;
```

Automatic Undo Management: UNDO Tablespace

Create the UNDO tablespace with the database by adding a clause in the CREATE DATABASE command:

```
CREATE DATABASE db01
. . .
UNDO TABLESPACE undo1
DATAFILE '/u01/oradata/undodb01.dbf' SIZE 20M
AUTOEXTEND ON
```

Or create it later by using the CREATE UNDO TABLESPACE command:

```
CREATE UNDO TABLESPACE undo1
DATAFILE '/u01/oradata/undodb01.dbf'
SIZE 20M;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: UNDO Tablespace

Automatic undo management requires an UNDO tablespace. More than one UNDO tablespace may exist in the database, but only one UNDO tablespace can be active.

You can create the UNDO tablespace with the database by adding a clause to the CREATE DATABASE statement.

During database creation, if the UNDO_MANAGEMENT parameter is set to AUTO and you omit the UNDO tablespace clause from the CREATE DATABASE statement then the Oracle server creates an UNDO tablespace with the name SYS_UNDOTBS. The default data file, for data file tablespace SYS_UNDOTS, will have the name 'dbu1<oracle_sid>.dbf'. It will be located in \$ORACLE_HOME/dbs. The size is operating system dependent. AUTOEXTEND is set to ON.

After database creation, you can create an UNDO tablespace using the CREATE UNDO TABLESPACE command.

Automatic Undo Management: UNDO Tablespace (continued)

Using Oracle Enterprise Manager to Create an UNDO Tablespace

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Select Create from the right-mouse menu.
3. Enter the filename and file size in the General tabbed page.
4. Select Undo in the Type region.
5. Enter storage information in the Storage tabbed page.
6. Click Create.

Create Tablespace - system@U621

General Storage

Name: UNDO1

Datafiles

File Name	File Directory	Size	MB
UNDO1.dbf	/home1/user621/O...	5	MB

Status

☒ Online ☐ Read Only

☐ Offline Normal

Type

☐ Permanent

☐ Temporary

☐ Set as Default Temporary Tablespace

☒ Undo

Create Cancel Show SQL Help

Automatic Undo Management: Altering an UNDO Tablespace

- The **ALTER TABLESPACE** command can make changes to UNDO tablespaces.
- The following example adds another data file to the UNDO tablespace:

```
ALTER TABLESPACE undotbs  
ADD DATAFILE '/u01/oradata/undotbs2.dbf'  
SIZE 30M  
AUTOEXTEND ON;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Altering an UNDO Tablespace

The server provides support for the following clauses when altering an UNDO tablespace.

- ADD DATAFILE
- RENAME
- DATAFILE [ONLINE | OFFLINE]
- BEGIN BACKUP
- END BACKUP

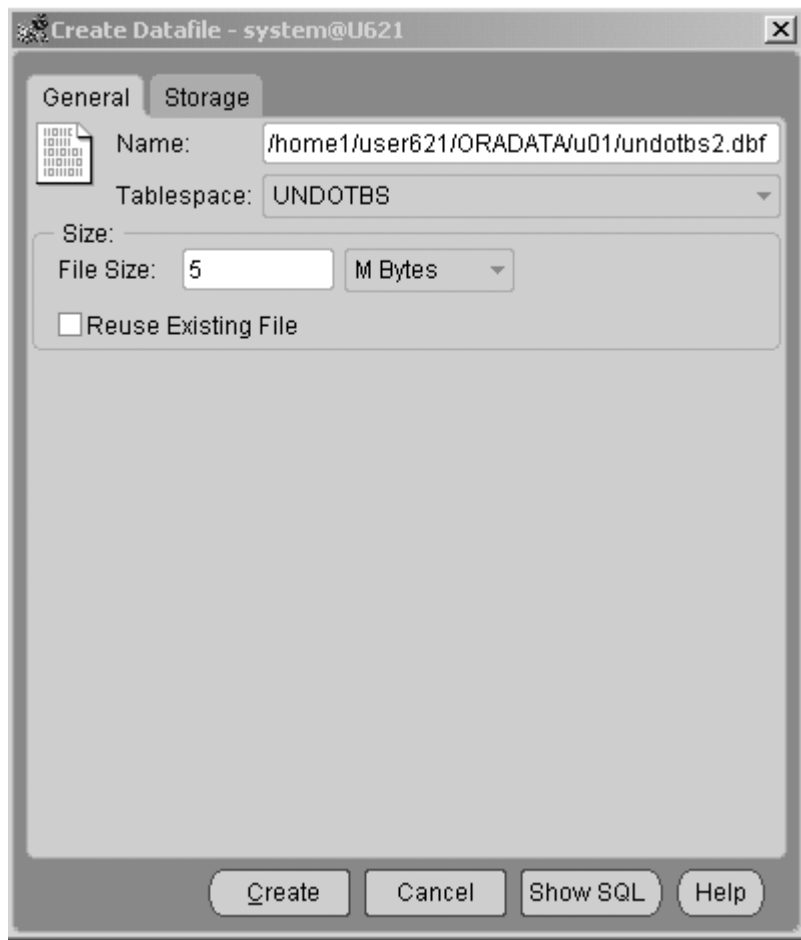
Automatic Undo Management: Altering an UNDO Tablespace (continued)

Using Oracle Enterprise Manager to Alter an UNDO Tablespace.

The following example adds a data file to the undo tablespace.

From the OEM Console:

1. Navigate to Storage > Tablespaces.
2. Highlight the undo tablespace name in which you want to add a data file.
3. Select Add a Datafile and complete the General and Storage tabbed page.
4. Click Create.



Automatic Undo Management: Switching UNDO Tablespaces

- You can switch from using one UNDO tablespace to another.
- Only one UNDO tablespace can be assigned to a database at a time.
- More than one UNDO tablespace may exist within an instance, but only one can be active.
- Use the **ALTER SYSTEM** command for dynamic switching between UNDO tablespaces.

```
ALTER SYSTEM SET UNDO_TABLESPACE=UNDOTBS2;
```

ORACLE

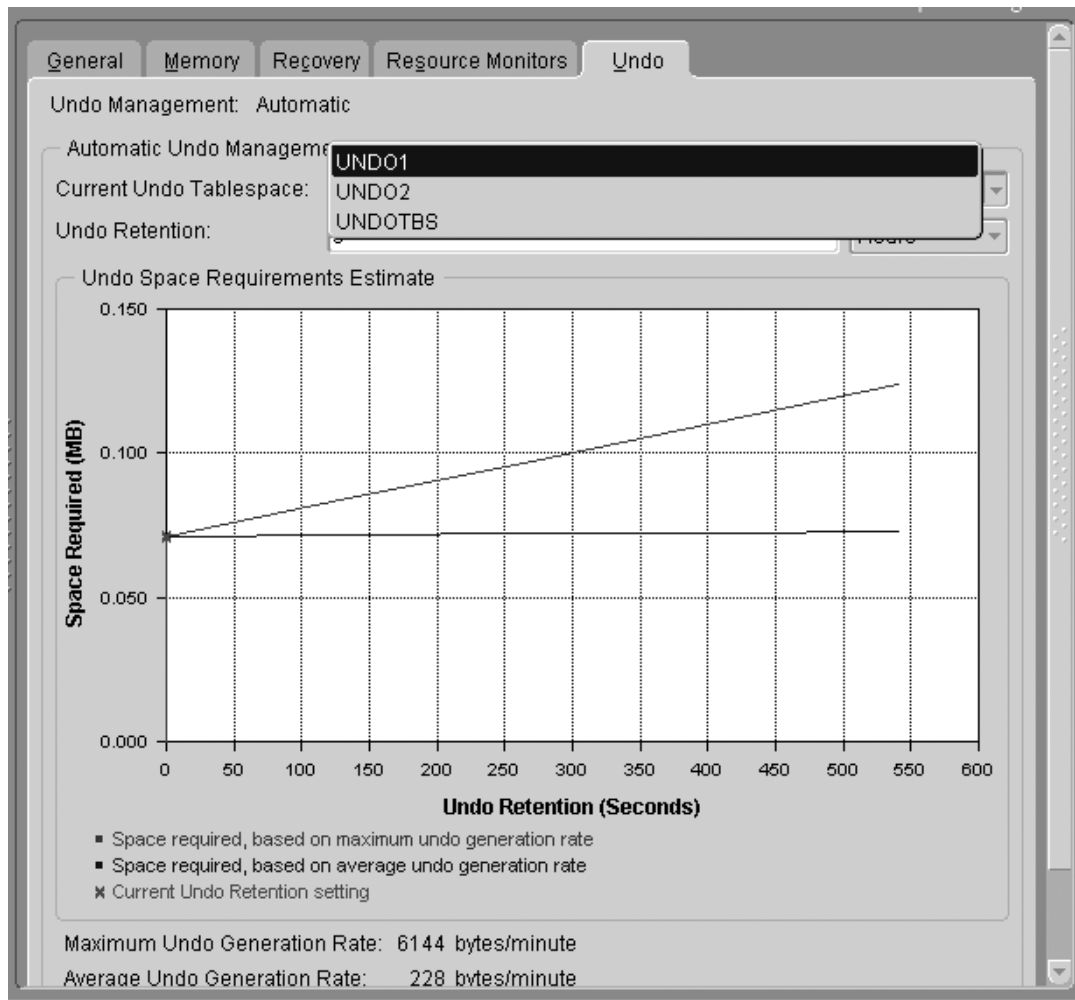
Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Switching UNDO Tablespaces

Using Oracle Enterprise Manager to Switch an UNDO Tablespace

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Select the Undo tabbed page.
2. Select the UNDO tablespace that you want to be current from the Current Undo tablespace drop-down list.
3. Click Apply.



Automatic Undo Management: Dropping an UNDO Tablespace

- The **DROP TABLESPACE** command drops an UNDO tablespace.

```
DROP TABLESPACE UNDOTBS2;
```

- An UNDO tablespace can only be dropped if it is currently not in use by any instance.
- To drop an active UNDO tablespace:
 - Switch to a new UNDO tablespace.
 - Drop the tablespace after all current transactions are complete.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Dropping an UNDO Tablespace

When dropping an UNDO tablespace, it cannot be in use by an instance and all transactions within the tablespace must be complete.

If tablespace UNDOTBS is the current active UNDO tablespace for the database, and it is to be dropped, then a new UNDO tablespace must be set before dropping it. First create another UNDO tablespace if one does not already exist. Then use the **ALTER SYSTEM** command to change the current UNDO tablespace.

```
SQL> ALTER SYSTEM SET undo_tablespace = UNDOTBS2;
```

You can drop tablespace UNDOTBS after all transactions within the tablespace are complete. To determine whether any active transactions exists use the following query:

```
SQL> SELECT a.name,b.status
2  FROM    v$rollname a, v$rollstat b
3  WHERE   a.name IN ( SELECT segment_name
4  FROM     dba_segments
5  AND a.usn = b.usn;
```

NAME	STATUS
-----	-----
_SYSSMU4\$	PENDING OFFLINE

Automatic Undo Management: Dropping an UNDO Tablespace (continued)

An undo segment with the status `PENDING OFFLINE` still contains active transactions. When no rows return from the query, then all transactions are complete, and the tablespace can be dropped with the following command.

```
SQL> DROP TABLESPACE UNDOTBS;
```

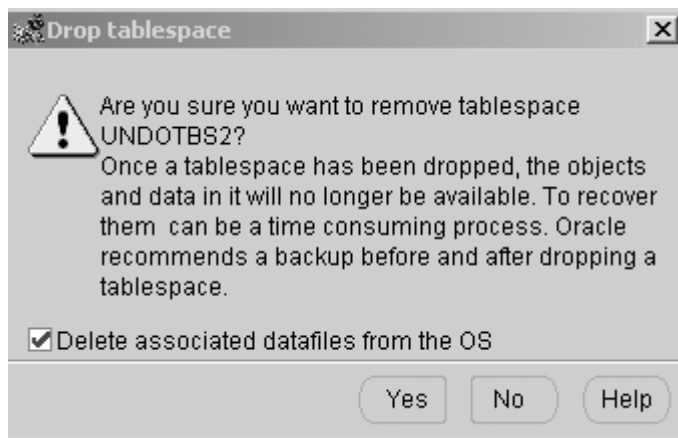
The Oracle server may reference tablespace `UNDOTBS` after switching to another UNDO tablespace to provide a consistent read for queries. Queries requiring information from tablespace `UNDOTBS`, after it is no longer available to provide a consistent read, receive the error: `ORA-1555 snapshot too old`.

Automatic Undo Management: Dropping an UNDO Tablespace (continued)

Using Oracle Enterprise Manager to Drop an UNDO Tablespace

From the OEM Console:

1. Navigate to Storage > Tablespaces
2. Highlight the UNDO tablespace you want to remove
3. Select Remove from the right mouse menu.
4. Select Delete associated datafiles from the OS.
5. Select Yes to confirm drop.



Automatic Undo Management: Other Parameters

- **UNDO_SUPPRESS_ERRORS parameter:**
 - Set to **TRUE**, this parameter suppresses errors while attempting to execute manual operations in **AUTO** mode.
- **UNDO_RETENTION parameter:**
 - This parameter controls the amount of undo data to retain for consistent read.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Other Parameters

UNDO_SUPPRESS_ERRORS Parameter

UNDO_SUPPRESS_ERRORS enables users to suppress errors while executing manual undo management mode operations (for example, `ALTER ROLLBACK SEGMENT ONLINE`) in automatic undo management mode. Setting this parameter enables users to use the undo tablespace feature before all application programs and scripts are converted to automatic undo management mode. For example, if you have an application that uses `SET TRANSACTION USE ROLLBACK SEGMENT` statement, you can add the statement `ALTER SESSION SET UNDO_SUPPRESS_ERRORS = true` to the application to suppress the ORA-30019 error.

ORA-30019: Illegal rollback segment operation in Automatic
Undo mode

UNDO_RETENTION Parameter

Determines how long to retain undo data to provide for consistent reads. Retaining undo data allows for longer queries and also requires larger data files for the UNDO tablespace. The UNDO_RETENTION parameter, defined in seconds, can be set in the initialization file or modified dynamically with an `ALTER SYSTEM` command.

```
SQL> ALTER SYSTEM SET UNDO_RETENTION=900;
```

A value of 900 retains undo data for 15 minutes.

Automatic Undo Management: Other Parameters (continued)

Even after you set `UNDO_RETENTION`, if the UNDO tablespace is sized too small, undo data is not retained for the time specified. The Oracle server uses an algorithm for allocating space within an UNDO tablespace and allocates unexpired space with no active transactions before causing a new transaction to fail.

Undo Data Statistics

```
SELECT end_time,begin_time,undoblks
FROM   v$undostat;
```

END_TIME	BEGIN_TIME	UNDO
22-JAN-01 13:44:18	22-JAN-01 13:43:04	19
22-JAN-01 13:43:04	22-JAN-01 13:33:04	1474
22-JAN-01 13:33:04	22-JAN-01 13:23:04	1347
22-JAN-01 13:23:04	22-JAN-01 13:13:04	1628
22-JAN-01 13:13:04	22-JAN-01 13:03:04	2249
22-JAN-01 13:03:04	22-JAN-01 12:53:04	1698
22-JAN-01 12:53:04	22-JAN-01 12:43:04	1433
22-JAN-01 12:43:04	22-JAN-01 12:33:04	1532
22-JAN-01 12:33:04	22-JAN-01 12:23:04	1075

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Undo Data Statistics

V\$UNDOSTAT view

This view displays a histogram of statistical data to show how well the database is working. Each row in the view keeps statistics collected in the instance for a 10-minute interval. You can use this view to estimate the amount of undo space required for the current workload. The Oracle server uses this to tune undo usage in the system. This view is available in both auto mode and manual mode.

Although the time interval is normally 10 minutes, the most recent row will return the time since its interval started, usually less than 10 minutes.

Automatic Undo Management: Sizing an UNDO Tablespace

Determining a size for the UNDO tablespace requires three pieces of information:

- **(UR) UNDO_RETENTION in seconds**
- **(UPS) Number of undo data blocks generated per second**
- **(DBS) Overhead varies based on extent and file size (db_block_size)**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Sizing an UNDO Tablespace

Sizing an UNDO tablespace requires three pieces of data. Two can be obtained from the initialization file: UNDO_RETENTION and DB_BLOCK_SIZE. The third piece of the formula requires a query against the database. The number of undo blocks generated per second can be acquired from V\$UNDOSTAT. The following formula calculates the total number of blocks generated and divides it by the amount of time monitored, in seconds:

```
SQL> SELECT (SUM(undoblks) / SUM(  
2  ((end_time - begin_time) * 86400)  
3  FROM v$undostat;
```

Column END_TIME and BEGIN_TIME are DATE data types. When DATE data types are subtracted, the result is in days. To convert days to seconds, you multiply by 86400, the number of seconds in a day.

The result of the query returns the number of undo blocks per second. This value must be multiplied by the size of an undo block, which is the same size as the database block defined in DB_BLOCK_SIZE. The following query calculates the number of bytes needed:

Automatic Undo Management: Sizing an UNDO Tablespace (continued)

```
SQL> SELECT (UR * (UPS * DBS)) + (DBS * 24) AS "Bytes"
 2  FROM (SELECT value AS UR
 3  FROM v$parameter
 4  WHERE name = 'undo_retention'),
 5  (SELECT (SUM(undoblks)/SUM
 6  (((end_time-begin_time)*86400))) AS UPS
 7  FROM v$undostat),
 8  (SELECT value AS DBS
 9  FROM v$parameter
10 WHERE name = 'db_block_size');

Bytes
-----
19106213
```

To convert bytes to megabytes, divide by 1,048,576 bytes. The result for this database is 18.22 MB.

For best results, the calculation should be made during the time of day when the database has its heaviest workload.

Automatic Undo Management: Undo Quota

- Long transactions and improperly written transactions can consume valuable resources.
- With undo quota, users can be grouped and a maximum undo space limit can be assigned to the group.
- UNDO_POOL, a Resource Manager directive, defines the amount of space allowed for a resource group.
- When a group exceeds its limit, no new transactions are possible for the group, until undo space is freed by current transactions which are either completing or aborting.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Automatic Undo Management: Undo Quota

Using resource plans, users can be grouped and limits placed on the amount of resources that can be used by the group. The amount of undo data generated by a group can be limited by setting a value for UNDO_POOL: the default value is unlimited. When a group exceeds its limit an error is received and no new transactions can be performed for the group until current transactions complete or abort.

ORA-30027: "Undo quota violation - failed to get %s (bytes)"

Cause: The amount of undo assigned to the consumer group of this session has been exceeded.

Action: Ask DBA to increase undo quota, or wait until other transactions commit before proceeding.

Note: Resource Management is discussed further in the "Managing Password Security and Resources" lesson.

Obtaining Undo Segment Information

- **Information about undo segments can be obtained by querying the following views:**
 - DBA_ROLLBACK_SEGS
- **Dynamic Performance Views**
 - V\$ROLLNAME
 - V\$ROLLSTAT
 - V\$UNDOSTAT
 - V\$SESSION
 - V\$TRANSACTION

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Obtaining Undo Segment Information

To obtain information about all the undo segments in the database, query the DBA_ROLLBACK_SEGS view:

```
SQL> SELECT segment_name,owner,tablespace_name,status
2 FROM dba_rollback_segs;
```

SEGMENT_NA	OWNER	TABLESPACE	STATUS
-----	-----	-----	-----
SYSTEM	SYS	SYSTEM	ONLINE
_SYSSMU1\$	PUBLIC	UNDO1	ONLINE
_SYSSMU2\$	PUBLIC	UNDO1	ONLINE
_SYSSMU3\$	PUBLIC	UNDO1	ONLINE
_SYSSMU4\$	PUBLIC	UNDO1	ONLINE

Information about undo segments that are offline can be seen only in DBA_ROLLBACK_SEGS. The dynamic performance views show only undo segments that are online.

The OWNER column specifies the type of an undo segment:

- SYS: Refers to a private undo segment
- PUBLIC: Refers to a public undo segment

Obtaining Undo Segment Information (continued)

V\$ROLLSTAT and V\$ROLLNAME Views

Join the V\$ROLLSTAT and V\$ROLLNAME views to obtain the statistics of the undo segments currently used by the instance.

Example

```
SQL> SELECT n.name, s.extents, s.rssize,s.hwmsize,
2      s.xacts, s.status
3      FROM v$rollname n, v$rollstat s
4      WHERE n.usn = s.usn;
```

NAME	EXTENTS	RSSIZE	HWMSIZE	XACTS	STATUS
-----	-----	-----	-----	-----	-----
SYSTEM	7	425984	425984	0	ONLINE
_SYSSMU1\$	5	2289664	2289664	0	ONLINE
_SYSSMU2\$	10	6549504	6549504	0	ONLINE
_SYSSMU3\$	7	4386816	4386816	0	ONLINE
_SYSSMU4\$	6	4321280	4321280	0	ONLINE

V\$TRANSACTION and V\$SESSION Views

To check the use of a undo segment by currently active transactions, join the V\$TRANSACTION and V\$SESSION views:

Example

```
SQL> SELECT s.username, t.xidusn, t.ubafil,
2      t.ubablk, t.used_ublk
3      FROM v$session s, v$transaction t
4      WHERE s.saddr = t.ses_addr;
```

USERNAME	XIDUSN	UBAFIL	UBABLK	USED_UBLK
-----	-----	-----	-----	-----
HR	2	2	5005	1

Summary

In this lesson, you should have learned how to:

- **Configure Automatic Undo Management**
- **Create an UNDO tablespace**
- **Properly size an UNDO tablespace**
- **Obtain undo segment information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 10 Overview

This practice covers the following topics:

- **Creating an UNDO tablespace**
- **Switching between UNDO tablespaces**
- **Dropping an UNDO tablespace**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 10 Overview

Note: This practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 10: Managing Undo Data

- 1 Connect as user `SYSTEM/MANAGER`, and list the undo segments in tablespace `UNDOTBS`.
- 2 Create undo tablespace `UNDO2`, size 15M, in `$HOME/ORADATA/u03`. List the undo segments in tablespace `UNDO2`.
- 3 In a new telnet session start `SQL*Plus` and connect as user `HR` and run script `lab10_03.sql` to insert a row into table `DEPARTMENTS`. Do not commit, roll back, or exit the session.
- 4 In the session in which you are connected as `SYS`, using the `ALTER SYSTEM` command, switch the `UNDO` tablespace from `UNDOTBS` to `UNDO2`, using `SCOPE=BOTH`.
- 5 As `SYS`, drop tablespace `UNDOTBS`. Use the `INCLUDING CONTENTS AND DATAFILES` clause. What happened?
- 6 List the undo segments in tablespace `UNDOTBS` and their status. Compare this list to the list in step 1.
Hint: Query `DBA_ROLLBACK_SEGS` data dictionary view.
- 7 In the session connected as `HR`, roll back the transaction and exit the session.
- 8 In the session connected as `SYS`, drop tablespace `UNDOTBS`. What happened?
- 9 As `SYS`, issue the following command:

```
ALTER SYSTEM SET undo_retention=0 SCOPE=memory;
```

Now drop tablespace `UNDOTBS`. What happened?

Note: There still may be a delay before the tablespace is dropped.

11

Managing Tables

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

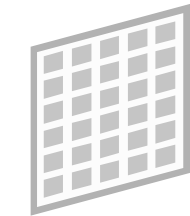
After completing this lesson, you should be able to do the following:

- **Identify the various methods of storing data**
- **Outline Oracle data types**
- **Distinguish between an extended versus a restricted ROWID**
- **Outline the structure of a row**
- **Create regular and temporary tables**
- **Manage storage structures within a table**
- **Reorganize, truncate, and drop a table**
- **Drop a column within a table**

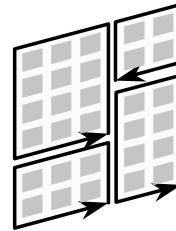
ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

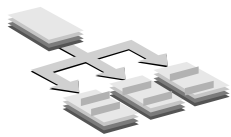
Storing User Data



Regular table



Partitioned table



Index-organized table



Cluster

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Storing User Data

There are several methods of storing user data in an Oracle database:

- Regular tables
- Partitioned tables
- Index-organized tables
- Clustered tables

Note: Partitioned tables, index-organized tables, and clustered tables are covered in other courses.

Regular Table

A regular table (generally referred to as a “table”) is the most commonly used form of storing user data. This is the default table and is the main focus of this lesson. A database administrator has very limited control over the distribution of rows in a table. Rows can be stored in any order depending on the activity in the table.

Storing User Data (continued)

Partitioned Table

A partitioned table enables the building of scalable applications. It has the following characteristics:

- A partitioned table has one or more partitions, each of which stores rows that have been partitioned using range partitioning, hash partitioning, composite partitioning, or list partitioning.
- Each partition in a partitioned table is a segment and can be located in a different tablespace.
- Partitions are useful for large tables that can be queried or manipulated using several processes concurrently.
- Special commands are available to manage partitions within a table.

Index-Organized Table

An index-organized table is like a heap table with a primary key index on one or more of its columns. However, instead of maintaining two separate storage spaces for the table and a B-tree index, an index-organized table maintains a single B-tree containing the primary key of the table and other column values. An overflow segment may exist due to the PCTTHRESHOLD value being set and the result of longer row lengths requiring the overflow area.

Index-organized tables provide fast key-based access to table data for queries involving exact matches and range searches.

Also, storage requirements are reduced because key columns are not duplicated in the table and index. The remaining non-key columns are stored in the index unless the index entry becomes very large; in that case, the Oracle server provides an OVERFLOW clause to handle the problem.

Storing User Data (continued)

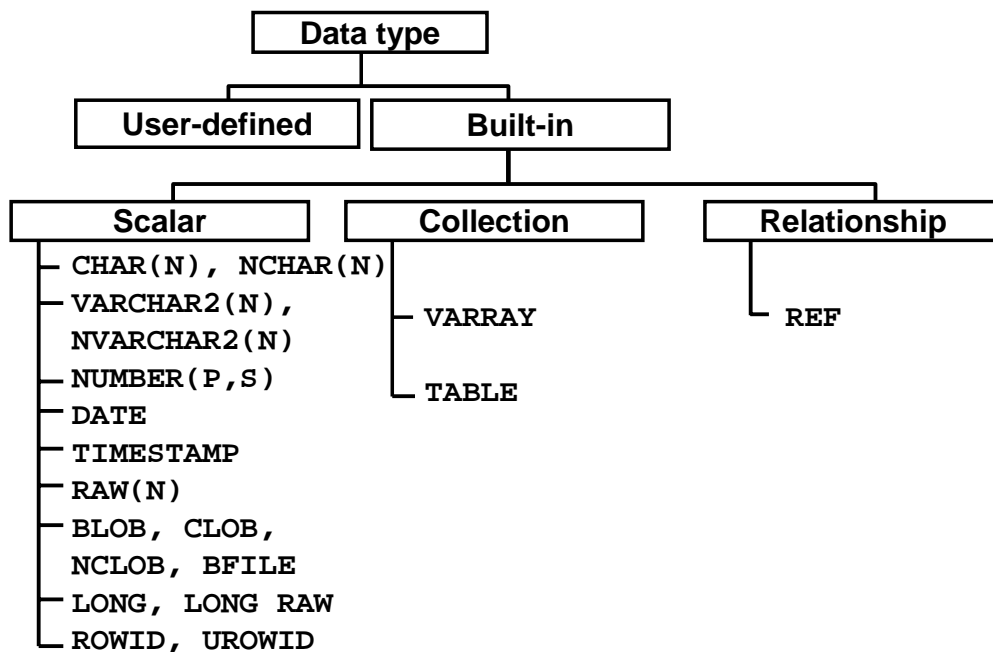
Clustered Table

A clustered table provides an optional method for storing table data. A cluster is made up of a table or group of tables that share the same data blocks, which are grouped together because they share common columns and are often used together.

Clusters have the following characteristics:

- Clusters have a *cluster key*, which is used to identify the rows that need to be stored together.
- The cluster key can consist of one or more columns.
- Tables in a cluster have columns that correspond to the cluster key.
- Clustering is a mechanism that is transparent to the applications using the tables. Data in a clustered table can be manipulated as though it were stored in a regular table.
- Updating one of the columns in the cluster key may migrate the row.
- The cluster key is independent of the primary key. The tables in a cluster can have a primary key, which may be the cluster key or a different set of columns.
- Clusters are usually created to improve performance. Random access to clustered data may be faster, but full table scans on clustered tables are generally slower.
- Clusters renormalize the physical storage of tables without affecting the logical structure.

Oracle Built-in Data Types



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Oracle Built-in Data Types

The Oracle server provides several built-in data types to store scalar data, collections, and relationships.

Scalar Data Types

Character data: Character data can be stored as either fixed-length or variable-length strings in the database.

Fixed-length character data types, such as CHAR and NCHAR, are stored with padded blanks. NCHAR is a Globalization Support data type that enables the storage of either fixed-width or variable-width character sets. The maximum size is determined by the number of bytes required to store one character, with an upper limit of 2,000 bytes per row. The default is one character or one byte, depending on the character set.

Variable-length character data types use only the number of bytes needed to store the actual column value, and can vary in size for each row, up to 4,000 bytes. VARCHAR2 and NVARCHAR2 are examples of variable-length character data types.

Oracle Built-in Data Types (continued)

Scalar Data Types (continued)

Numeric data type: Numbers in an Oracle database are always stored as variable-length data. They can store up to 38 significant digits. Numeric data types require:

- One byte for the exponent
- One byte for every two significant digits in the mantissa
- One byte for negative numbers if the number of significant digits is less than 38 bytes

DATE data type: The Oracle server stores dates in fixed-length fields of seven bytes. An Oracle DATE always includes the time.

TIMESTAMP data type: This data type stores the date and time including fractional seconds up to nine decimal places. `TIMESTAMP WITH TIME ZONE` and `TIMESTAMP WITH LOCAL TIME ZONE` can use time zones to factor items such as daylight savings time. `TIMESTAMP` and `TIMESTAMP WITH LOCAL TIME ZONE` can be used in primary keys, `TIMESTAMP WITH TIME ZONE` cannot.

RAW data type: This data type enables the storage of small binary data. The Oracle server does not perform character set conversion when RAW data is transmitted across machines in a network or if RAW data is moved from one database to another using Oracle utilities. The number of bytes needed to store the actual column value, and can vary in size for each row, up to 2,000 bytes.

LONG, LONG RAW, and Large Object (LOBs) Data Types

Oracle provides six data types for storing LOBs:

- CLOB and LONG for large fixed-width character data
- NCLOB for large fixed-width national character set data
- BLOB and LONG RAW for storing unstructured data
- BFILE for storing unstructured data in operating system files

LONG and LONG RAW data types were previously used for unstructured data, such as binary images, documents, or geographical information, and are primarily provided for backward compatibility. These data types are superseded by the LOB data types. LOB data types are distinct from LONG and LONG RAW, and they are not interchangeable. LOBs will not support the LONG application programming interface (API), and vice versa.

Oracle Built-in Data Types (continued)

LONG, LONG RAW, and Large Object (LOBs) Data Types (continued)

It is beneficial to discuss LOB functionality in comparison to the older types (LONG and LONG RAW). Below, LONGs refer to LONG and LONG RAW, and LOBs refer to all LOB data types.

LONG, LONG RAW	LOB
Single column per table	Multiple columns per table
Up to 2 gigabytes	Up to 4 gigabytes
Data stored in-line	Data stored in-line or out-of-line
No object type support	Supports object types
Sequential access to chunk	Random access to chunks

LOBs store a locator in the table and the data elsewhere, unless the size is less than the maximum size for a VARCHAR2 data type, which is 4,000 bytes; LONGs store all data in-line. In addition, LOBs allow data to be stored in a separate segment and tablespace, or in a host file.

LOBs support object type attributes (except NCLOBs) and replication; LONGs do not.

LONGs are primarily stored as chained row pieces, with a row piece in one block pointing to the next row piece stored in another block. Therefore, they need to be accessed sequentially. In contrast, LOBs support random piecewise access to the data through a file-like interface.

ROWID and UROWID Data Types

ROWID is a data type that can be queried along with other columns in a table. It has the following characteristics:

- ROWID is a unique identifier for each row in the database.
- ROWID is not stored explicitly as a column value.
- Although the ROWID does not directly give the physical address of a row, it can be used to locate the row.
- ROWID provides the fastest means of accessing a row in a table.
- ROWIDs are stored in indexes to specify rows with a given set of key values.

With release Oracle8.1, the Oracle server provides a single data type called the universal ROWID or UROWID. It supports ROWIDs of foreign tables (non-Oracle tables) and can store all kinds of ROWIDs. For example: A UROWID datatype is required to store a ROWID for rows stored in an index-organized table (IOT). The value of the parameter COMPATIBLE must be set to Oracle8.1 or higher to use UROWID.

Oracle Built-in Data Types (continued)

Collection Data Types

Two types of collection data types are available to store data that is repetitive for a given row in a table. Prior to Oracle8i, the Objects option was needed to define and use collections. A brief discussion of these types follows.

Varying arrays (VARRAY): Varying arrays are useful to store lists that contain a small number of elements, such as phone numbers for a customer.

VARRAYs have the following characteristics:

- An array is an ordered set of data elements.
- All elements of a given array are of the same data type.
- Each element has an index, which is a number corresponding to the position of the element in the array.
- The number of elements in an array determines the size of the array.
- The Oracle server allows arrays to be of variable size, which is why they are called VARRAYs, but the maximum size must be specified when declaring the array type.

Nested tables: Nested tables provide a means of defining a table as a column within a table. They can be used to store sets that may have a large number of records such as number of items in an order.

Nested tables generally have the following characteristics:

- A nested table is an unordered set of records or rows.
- The rows in a nested table have the same structure.
- Rows in a nested table are stored separate from the parent table with a pointer from the corresponding row in the parent table.
- Storage characteristics for the nested table can be defined by the database administrator.
- There is no predetermined maximum size for a nested table.

Relationship Data Types (REFs)

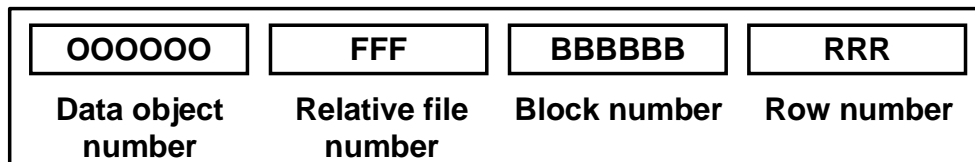
Relationship types are used as pointers within the database. The use of these types requires the Objects option. As an example, each item that is ordered could point to or reference a row in the PRODUCTS table, without having to store the product code.

Oracle User-Defined Data Types

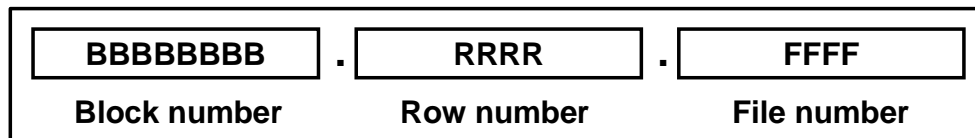
The Oracle server allows a user to define abstract data types and use them within the application.

ROWID Format

- **Extended ROWID Format**



- **Restricted ROWID Format**



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

ROWID Format

An extended ROWID needs 10 bytes of storage on disk and is displayed by using 18 characters. It consists of the following components:

- **Data object number:** Is assigned to each data object, such as table or index when it is created, and it is unique within the database
- **Relative file number:** Is unique to each file within a tablespace
- **Block number:** Represents the position of the block, containing the row, within the file
- **Row number:** Identifies the position of the row directory slot in the block header

Internally, the data object number needs 32 bits, the relative file number needs 10 bits, block number needs 22 bits, and the row number needs 16 bits, adding up to a total of 80 bits or 10 bytes.

An extended ROWID is displayed using a base-64 encoding scheme, which uses six positions for the data object number, three positions for the relative file number, six positions for the block number, and three positions for the row number. The base-64 encoding scheme uses characters A-Z, a-z, 0-9, and /. This is a total of 64 characters, as in the following example:

ROWID Format (continued)

```
SQL> SELECT department_id, rowid
       2 FROM hr.departments;
```

DEPARTMENT_ID	ROWID
10	AAABQMAAFAAAAA6AAA
20	AAABQMAAFAAAAA6AAB
30	AAABQMAAFAAAAA6AAC
40	AAABQMAAFAAAAA6AAD
50	AAABQMAAFAAAAA6AAE
60	AAABQMAAFAAAAA6AAF

...

In this example:

- AAABQM is the data object number.
- AAF is the relative file number.
- AAAAA6 is the block number.
- AAA is the row number for the department with ID = 10.

Restricted ROWID in Oracle7 and Earlier

Versions of the Oracle database prior to Oracle8 used the restricted ROWID format. A restricted ROWID used only six bytes internally and did not contain the data object number. This format was acceptable in Oracle7 or an earlier release because the file numbers were unique within a database. Thus, earlier releases did not permit more than 1,022 data files. Now it is the limit for a tablespace.

Even though Oracle8 removed this restriction by using tablespace-relative file numbers, the restricted ROWID is still used in objects like nonpartitioned indexes on nonpartitioned tables where all the index entries refer to rows within the same segment.

Locating a Row Using ROWID

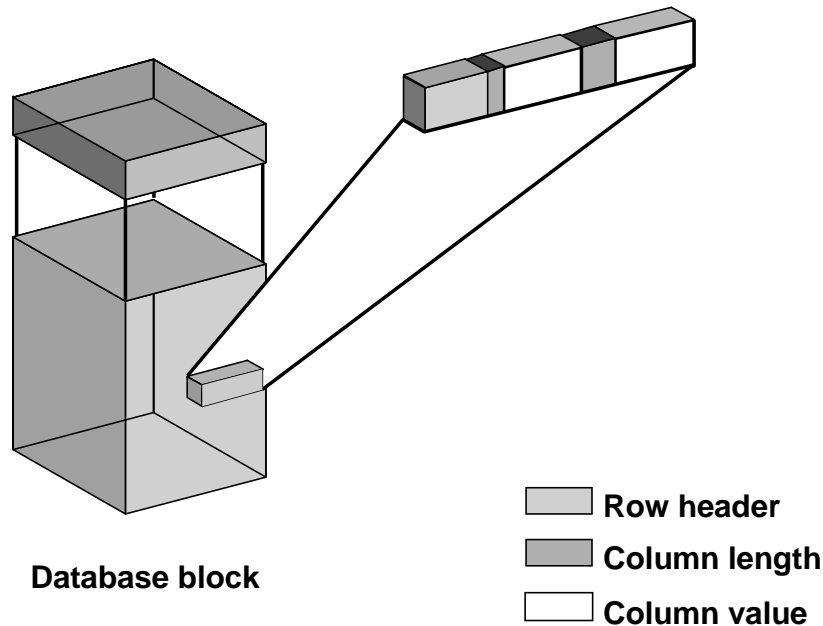
Because a segment can only reside in one tablespace, by using the data object number, the Oracle server can determine the tablespace that contains a row.

The relative file number within the tablespace is used to locate the file, the block number is used to locate the block containing the row, and the row number is used to locate the row directory entry for the row.

The row directory entry can be used to locate the beginning of the row.

Thus, ROWID can be used to locate any row within a database.

Structure of a Row



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Structure of a Row

Row data is stored in database blocks as variable-length records. Columns for a row are generally stored in the order in which they are defined and any trailing NULL columns are not stored.

Note: A single byte for column length is required for non trailing NULL columns. Each row in a table has:

- **Row header:** Used to store the number of columns in the row, the chaining information, and the row lock status
- **Row data:** For each column, the Oracle server stores the column length and value (One byte is needed to store the column length if the column will require more than 250 bytes of storage in which case three bytes will be used for column length. The column value is stored immediately following the column length bytes.)

Adjacent rows do not need any space between them. Each row in the block has a slot in the row directory. The directory slot points to the beginning of the row.

Creating a Table

```
CREATE TABLE hr.employees(  
    employee_id NUMBER(6),  
    first_name VARCHAR2(20),  
    last_name VARCHAR2(25),  
    email VARCHAR2(25),  
    phone_number VARCHAR2(20),  
    hire_date DATE DEFAULT SYSDATE,  
    job_id VARCHAR2(10),  
    salary NUMBER(8,2),  
    commission_pct NUMBER(2,2),  
    manager_id NUMBER(6),  
    department_id NUMBER(4))  
TABLESPACE USERS;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Table

The `CREATE TABLE` command is used to create relational tables or object tables.

Relational table: This is the basic structure to hold user data.

Object table: Is a table that uses an object type for a column definition. An object table is a table that is explicitly defined to hold the object instance of a particular type.

Note: Object tables will not be covered within this lesson.

Creating Table Guidelines

- Place tables in separate tablespaces.
- Use locally-managed tablespaces to avoid fragmentation.

Note: Refer to the *Oracle9i SQL Reference* document for more information on the various clauses and parameters that can be defined when using the `CREATE TABLE` command.

To create a relational table in your own schema, you must have the `CREATE TABLE` system privilege. To create a table in another user's schema, you must have `CREATE ANY TABLE` system privilege.

Note: Refer to the "Managing Privileges" lesson for details regarding granting privileges.

Creating a Table (continued)

The example below creates a DEPARTMENTS table in a data dictionary managed tablespace.

```
SQL> CREATE TABLE hr.departments(  
  2  department_id NUMBER(4),  
  3  department_name VARCHAR2(30),  
  4  manager_id NUMBER(6),  
  5  location_id NUMBER(4))  
  6  STORAGE(INITIAL 200K NEXT 200K  
  7  PCTINCREASE 0 MINEXTENTS 1 MAXEXTENTS 5)  
  8  TABLESPACE data;
```

The above syntax is a subset of the CREATE TABLE clause.

STORAGE Clause

The STORAGE clause specifies storage characteristics for the table. The storage allocated for the first extent is 200 KB. When a second extent is required it will be created at 200 KB also defined by the NEXT value. When a third extent is required, it will be created at 200 KB because the PCTINCREASE has been set to zero. The maximum amount of extents that can be used is set at five, with the minimum set to one.

- MINEXTENTS: This is the minimum number of extents that is to be allocated.
- MAXEXTENTS: This is the maximum number of extents to be allocated. If MINEXTENTS is specified with a value greater than one and the tablespace contains more than one datafile, the extents will be spread across the different data files.
- PCTINCREASE: This is the percent of increase in extent size after NEXT extent and thereafter.

Creating a Table (continued)

Block utilization parameters can also be specified in the physical attributes clause for the table.

- **PCTFREE**: Specifies the percentage of space in each data block of the table. The value of **PCTFREE** must be a value from zero to ninety-nine. A value of zero means that the entire block can be filled by inserts of new rows. The default value is 10. This value reserves 10% of each block for updates to existing rows and allows inserts of new rows to fill a maximum of 90% of each block.
- **PCTUSED**: Specifies the minimum percentage of used space that is maintained for each data block of the table. A block becomes a candidate for row insertion when its used space falls below **PCTUSED**. **PCTUSED** is specified as integer from zero to ninety-nine and defaults to 40.

The combination of **PCTFREE** and **PCTUSED** determines whether new rows will be inserted into existing data blocks or into new blocks. The sum of these two must be equal to or less than 100. These parameters are used to utilize space within a table more efficiently.

Note: **PCTUSED**, **FREELISTS**, and **FREELIST GROUPS** are deprecated with the Oracle9i feature Automatic Segment-Space Management. Refer to the “Storage Structures and Relationship” lesson for details regarding this feature.

- **INITTRANS**: Specifies the initial number of transaction entries allocated within each data block allocated to the table. This value can range from 1-255 and default to one. **INITTRANS**: Ensures that a minimum number of concurrent transactions can update the block. In general, this value should not be changed from its default.
- **MAXTRANS**: Specifies the maximum number of concurrent transaction that can update a data block allocated to the table. This limit does not apply to queries. The value can range from 1-255 and the default is a function of the data block size.

TABLESPACE Clause

The **TABLESPACE** clause specifies the tablespace where the table will be created. The table in the example will reside within the **data** tablespace. If you omit **TABLESPACE**, then Oracle creates the object in the default tablespace of the owner of the schema containing the table.

Note: Refer to the “Managing Tablespaces” lesson for more information regarding tablespaces.

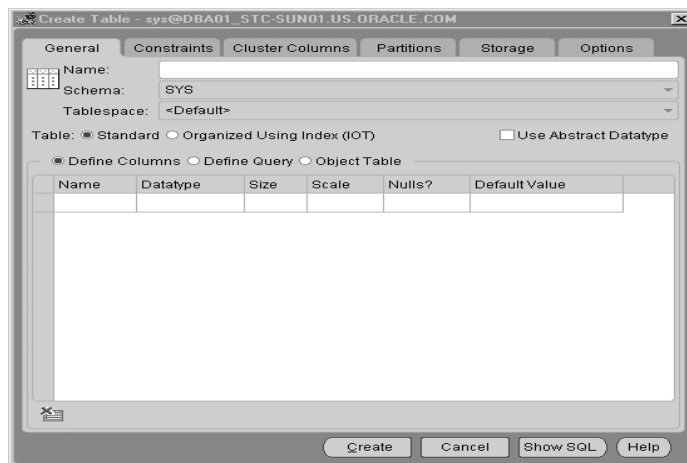
Creating a Table (continued)

Using Oracle Enterprise Manager to Create a Table

From the OEM Console:

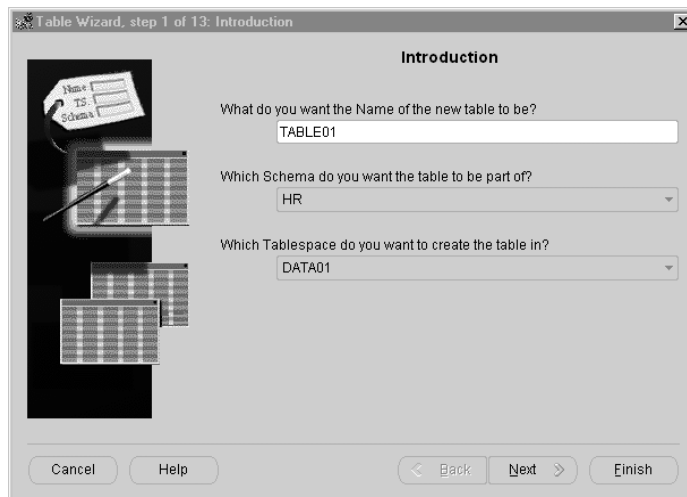
Note: OEM offers many options to creating a table.

1. Navigate to Schema > [Schema Name] > Tables.
2. Select Create from the right-mouse menu.
3. Enter your table information within the various pages.
4. Click Create.



Creating a table using the wizard:

1. Navigate to Schema > [Schema Name] > Tables.
2. Select Create Using Wizard from the right mouse menu.
3. Enter your table information.
4. Click Finish.



Creating a Table: Guidelines

- **Place tables in separate tablespaces.**
- **Use locally-managed tablespaces to avoid fragmentation.**
- **Use few standard extent sizes for tables to reduce tablespace fragmentation.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Table: Guidelines

Place tables in separate tablespaces, not in the tablespace that has undo segments, temporary segments, and indexes.

Place tables in locally managed tablespaces to avoid fragmentation.

Creating Temporary Tables

- **Created using the GLOBAL TEMPORARY clause:**

```
CREATE GLOBAL TEMPORARY TABLE  
hr.employees_temp  
AS SELECT * FROM hr.employees;
```

- **Tables retain data only for the duration of a transaction or session.**
- **DML locks are not acquired on the data.**
- **You can create indexes, views, and triggers on temporary tables.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating Temporary Tables

Temporary tables can be created to hold session-private data that exists only for the duration of a transaction or session.

The `CREATE GLOBAL TEMPORARY TABLE` command creates a temporary table that can be transaction-specific or session-specific. For transaction-specific temporary tables, data exists for the duration of the transaction, while for session-specific temporary tables, data exists for the duration of the session. Data in a session is private to the session. Each session can see and modify only its own data. DML locks are not acquired on the data of the temporary tables. The clauses that control the duration of the rows are:

- `ON COMMIT DELETE ROWS`: To specify that rows are only visible within the transaction. This is the default.
- `ON COMMIT PRESERVE ROWS`: To specify that rows are visible for the entire session.

You can create indexes, views, and triggers on temporary tables and you can also use the Export and Import utilities to export and import the definition of a temporary table. However, no data is exported, even if you use the `ROWS` option. The definition of a temporary table is visible to all sessions.

Setting PCTFREE and PCTUSED

- **Compute PCTFREE**

$$\frac{(\text{Average Row Size} - \text{Initial Row Size}) * 100}{\text{Average Row Size}}$$

- **Compute PCTUSED**

$$100 - \text{PCTFREE} - \frac{\text{Average Row Size} * 100}{\text{Available Data Space}}$$

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Setting PCTFREE and PCTUSED

Setting PCTFREE

A higher PCTFREE affords more room for updates within a database block. Set a higher value if the table contains:

- Columns that are initially NULL and later updated with a value
- Columns that are likely to increase in size as a result of an update

A higher PCTFREE will result in lower block density—each block can accommodate fewer rows.

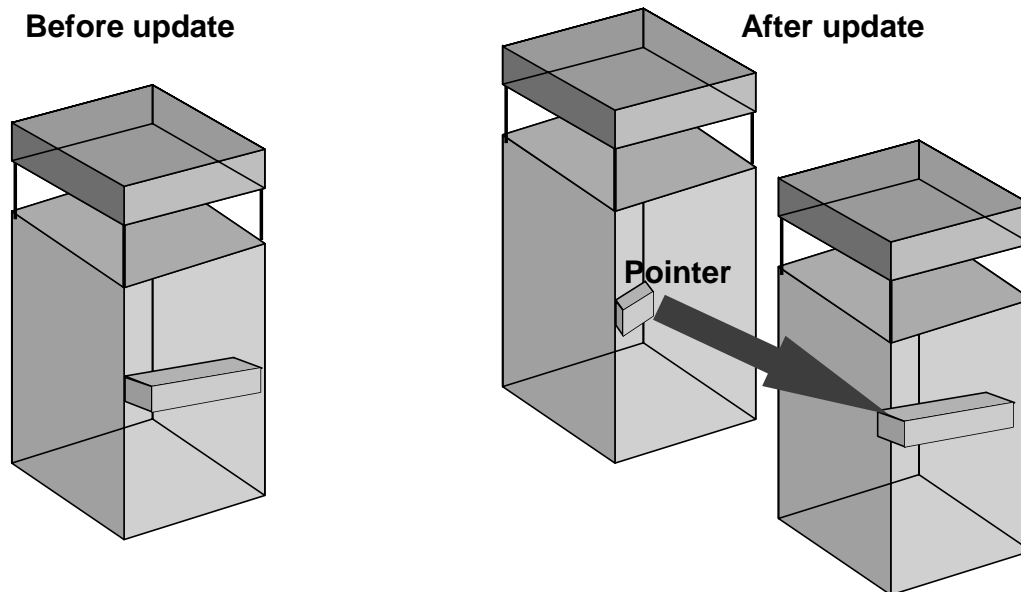
The formula specified above ensures that there is enough free space in the block for row growth.

Setting PCTUSED

Set PCTUSED to ensure that the block is returned to the free list only when there is sufficient space to accommodate an average row. If a block on the free list does not contain sufficient space for inserting a row, the Oracle server looks up the next block on the free list. This linear scan continues until either a block with sufficient space is found or the end of the list is reached. Using the formula given reduces the time taken to scan the free list by increasing the probability of finding a block with the required free space.

Note: The value for average row size can be estimated using the `ANALYZE TABLE` command.

Row Migration and Chaining



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Row Migration and Chaining

Row Migration

If PCTFREE is set to a low value, there may be insufficient space in a block to accommodate a row that grows as a result of an update. When this happens, the Oracle server will move the entire row to a new block and leave a pointer from the original block to the new location. This process is referred to as row migration. When a row is migrated, input/output (I/O) performance associated with this row decreases because the Oracle server must scan two data blocks to retrieve the data.

Row Chaining

Row chaining occurs when a row is too large to fit into any block. This might occur when the row contains columns that are very long. In this case, the Oracle server divides the row into smaller chunks called row pieces. Each row piece is stored in a block along with the necessary pointers to retrieve and assemble the entire row. Row chaining can be minimized by choosing a higher block size or by splitting the table into multiple tables with fewer columns, if possible.

Note: Row migration and row chaining are covered in more detail in the *Oracle9i Database Performance Tuning* course.

Changing Storage and Block Utilization Parameters

```
ALTER TABLE hr.employees
PCTFREE 30
PCTUSED 50
STORAGE(NEXT 500K
MINEXTENTS 2
MAXEXTENTS 100);
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Changing Storage and Block Utilization Parameters

Some of the storage parameters and any of the block utilization parameters can be modified by using the `ALTER TABLE` command.

Syntax

```
ALTER TABLE [schema.]table
{[ storage-clause ]
[ INITTRANS integer ]
[ MAXTRANS integer]}
```

Effects of Changing Storage Parameters

The parameters that can be modified and the implications of the modifications are as follows:

- **NEXT:** When the Oracle server allocates another extent for the table, the new value will be used. Subsequent extent sizes will increase by `PCTINCREASE`.

Changing Storage and Block Utilization Parameters (continued)

Effects of Changing Storage Parameters (continued)

- **PCTINCREASE:** A change in PCTINCREASE is registered in the data dictionary. It is used to recalculate NEXT when the next extent is allocated by the Oracle server.
Consider a case where a table with two extents has NEXT=10K and PCTINCREASE=0. If PCTINCREASE is changed to 100, the third extent to be allocated will be 10 KB, the fourth extent will be 20 KB, the fifth extent will be 40 KB, and so on.
- **MINEXTENTS:** The value of MINEXTENTS can be changed to any value that is less than or equal to the current number of extents in the table. It will have no immediate effect on the table, but will be used if the table is truncated.
- **MAXEXTENTS:** The value of MAXEXTENTS can be set to any value equal to or greater than the current number of extents for the table. The value can also be set to UNLIMITED.

Restrictions

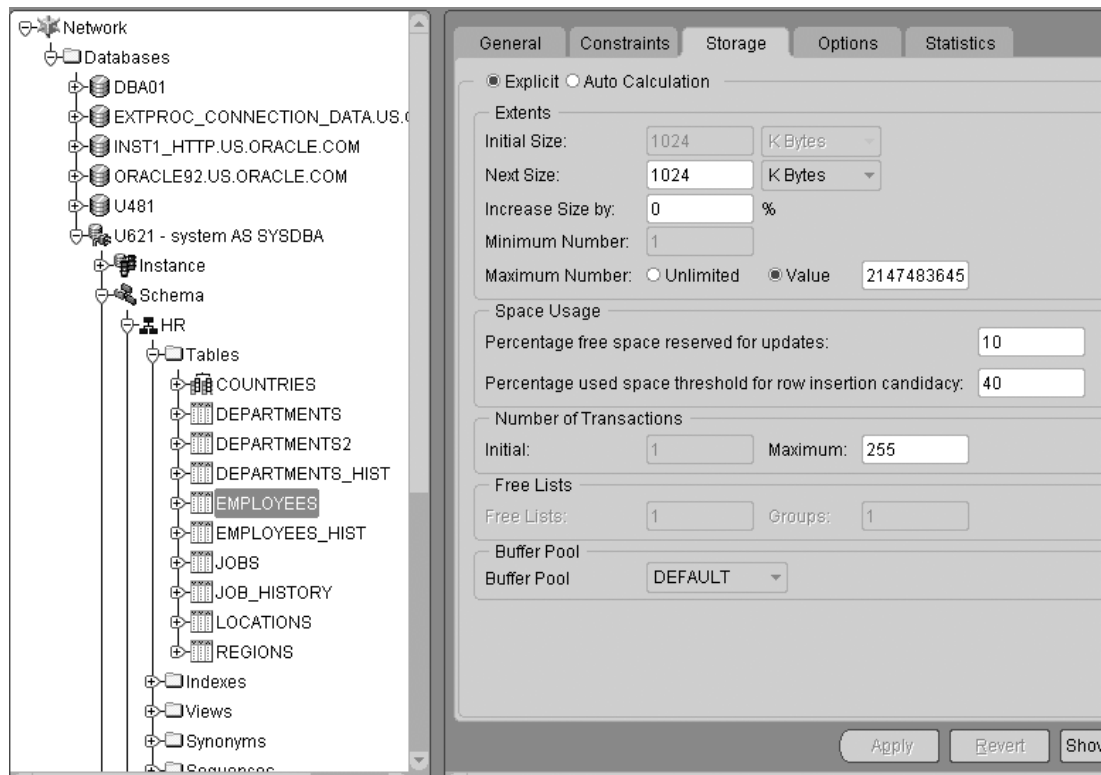
- The value of INITIAL cannot be modified for a table.
- The value of NEXT specified will be rounded to a value that is a multiple of the block size greater than or equal to the value specified.

Changing Storage and Block Utilization Parameters (continued)

Using Oracle Enterprise Manager to Change Storage Parameters

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Tables.
2. Highlight the table.
3. Select the Storage tabbed page.
4. Modify the values in the Storage tabbed page. Note that the minimum extents and initial number of transactions cannot be modified using this method.
5. Click Apply.



Manually Allocating Extents

```
ALTER TABLE hr.employees  
ALLOCATE EXTENT(SIZE 500K  
DATAFILE '/DISK3/DATA01.DBF');
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Manually Allocating Extents

Extents may need to be allocated manually:

- To control the distribution of extents of a table across files
- Before loading data in bulk to avoid dynamic extension of tables

Syntax

Use the following command to allocate an extent to a table:

```
ALTER TABLE [schema.]table  
  ALLOCATE EXTENT [ ([SIZE integer [K|M]]  
  [ DATAFILE 'filename' ]) ]
```

If SIZE is omitted, the Oracle server will use the NEXT_EXTENT size from DBA_TABLES to allocate the extent.

The file specified in the DATAFILE clause must belong to the tablespace that the table belongs to. Otherwise, the statement will generate an error. If the DATAFILE clause is not used, the Oracle server will allocate the extent in one of the files in the tablespace containing the table.

Note: The NEXT_EXTENT value in DBA_TABLES will not be affected by manual extent allocation. The Oracle server will not recalculate the size of the next extent when this command is executed.

Nonpartitioned Table Reorganization

```
ALTER TABLE hr.employees  
MOVE TABLESPACE data1;
```

- When a nonpartitioned table is reorganized, its structure is kept, but not its contents.
- It is used to move a table to a different tablespace or reorganize extents.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Nonpartitioned Table Reorganization

A nonpartitioned table can be moved without having to run the Export or Import utility. In addition, it allows the storage parameters to be changed. This is useful when:

- Moving a table from one tablespace to another
- Reorganizing the table to eliminate row migration

After moving a table you must rebuild the indexes to avoid the following error:

```
SQL> SELECT * FROM employees WHERE id=23;
```

```
select * from employees where id=23
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01502: index 'HR.EMPLOYEES_ID_PK' or partition of  
such index is in unusable state
```

Truncating a Table

```
TRUNCATE TABLE hr.employees;
```

- **Truncating a table deletes all rows in a table and releases used space.**
- **Corresponding indexes are truncated.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Truncating a Table

Syntax

```
TRUNCATE TABLE [schema.] table  
[ {DROP | REUSE} STORAGE]
```

The effects of using this command are as follows:

- All rows in the table are deleted.
- No undo data is generated and the command commits implicitly because TRUNCATE TABLE is a DDL command.
- Corresponding indexes are also truncated.
- A table that is being referenced by a foreign key cannot be truncated.
- The delete triggers do not fire when this command is used.

Dropping a Table

```
DROP TABLE hr.departments  
CASCADE CONSTRAINTS;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dropping a Table

A table can be dropped if it is no longer needed or if it is to be reorganized.

Syntax

Use the following command to drop a table:

```
DROP TABLE [schema.] table  
[CASCADE CONSTRAINTS]
```

When a table is dropped, the extents used by the table are released. If they are contiguous, they may be coalesced either automatically or manually at a later stage.

The CASCADE CONSTRAINTS option is necessary if the table is the parent table in a foreign key relationship.

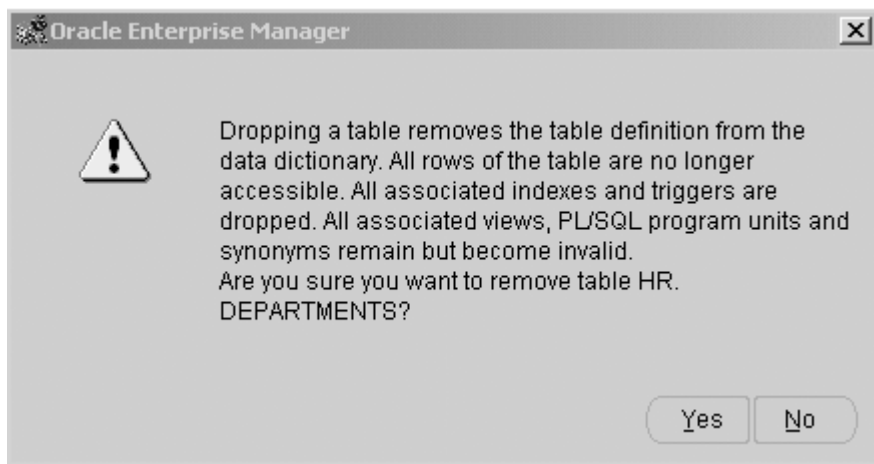
Note: Refer to the “Maintaining Data Integrity” lesson for details regarding CASCADE CONSTRAINTS.

Dropping a Table (continued)

Using Oracle Enterprise Manager to Drop a Table

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Tables.
2. Highlight the table.
3. Select Remove from the right-mouse menu.
4. Select Yes to confirm drop.



Dropping a Column

Removing a column from a table:

```
ALTER TABLE hr.employees  
DROP COLUMN comments  
CASCADE CONSTRAINTS CHECKPOINT 1000;
```

- Removes the column length and data from each row, freeing space in the data block.
- Dropping a column in a large table takes a considerable amount of time.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dropping a Column

The Oracle server enables you to drop columns from rows in a table. Dropping columns cleans unused and potentially space-demanding columns without having to export or import data, and recreate indexes and constraints.

Dropping a column can take a significant amount of time because all the data for the column is deleted from the table.

Before Oracle8i, it was not possible to drop a column from a table.

Using a Checkpoint when Dropping a Column

Dropping a column can be time consuming and requires a large amount of undo space. While dropping columns from large tables, checkpoints can be specified to minimize the use of undo space. In the example in the slide, a checkpoint occurs every 1,000 rows. The table is marked `INVALID` until the operation completes. Refer to the `STATUS` column in the `DBA_OBJECTS` view. If the instance fails during the operation, the table remains `INVALID` on start up, and the operation will have to be completed.

Use the following statement to resume an interrupted drop operation:

```
SQL> ALTER TABLE hr.employees DROP COLUMNS CONTINUE;
```

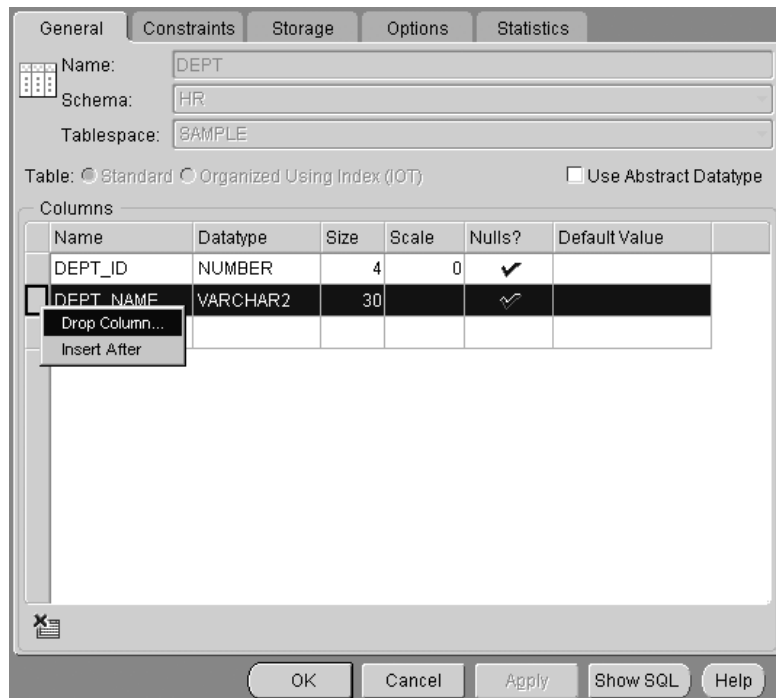
Use of this will generate an error if the table is in a `VALID` state.

Dropping a Column (continued)

Using Oracle Enterprise Manager to Drop a Column

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Tables.
2. Highlight the table.
3. Select Edit/View Details from the right-mouse menu.
4. Select the column to be dropped.
5. Select Drop Column from right-mouse menu.



Renaming a Column

Renaming a column from a table:

```
ALTER TABLE hr.employees  
RENAME COLUMN hire_date  
TO start_date;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Renaming a Column

In Oracle9i Database Release 2, the functionality for renaming columns is available for columns belonging to a relational table. Columns cannot be renamed for tables with join indexes, the indexes must first be dropped before renaming is allowed.

After renaming a column, functional indexes and check constraints would remain valid, however views, triggers, domain indexes, functions, procedures and packages would be invalidated. If the revalidation of the above fails due to column name changing, you must resolve the issue with the new name.

Renaming is allowed for tables with materialized views and tables involved in replication. If errors in the materialized views occur subsequently, you must modify the materialized views to work out the issues. The syntax is:

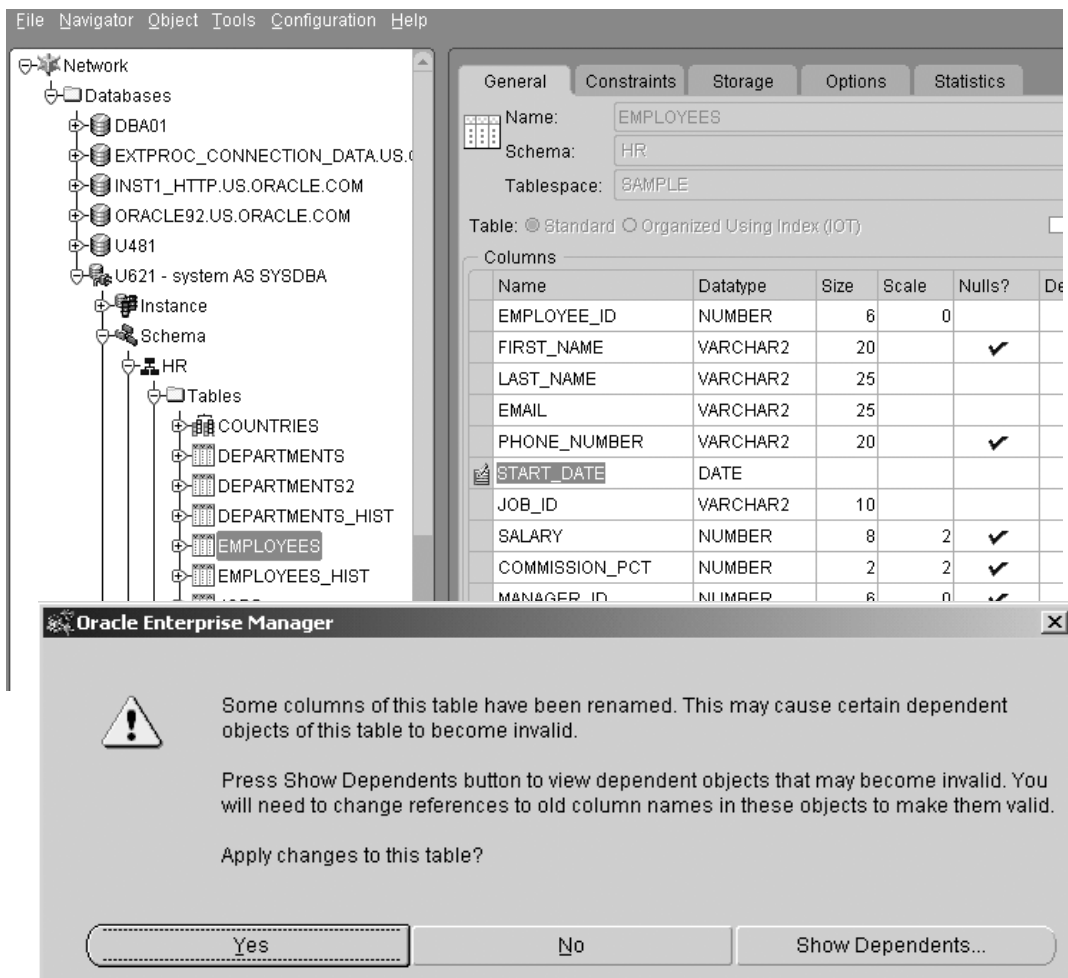
```
SQL> ALTER TABLE [schema.]table_name  
2  RENAME COLUMN old_column_name  
3  TO new_column_name;
```

Renaming a Column (continued)

Using Oracle Enterprise Manager to Rename a Column

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Tables
2. Highlight the table containing the column you want to rename.
3. Select the column you want to change.
4. Enter the name of the new column name over the old.
5. Click Apply.
6. Select Show Dependents to check dependencies on the column name.
7. Select Yes if the name change will not affect dependencies.



Using the UNUSED Option

- **Mark a column as unused:**

```
ALTER TABLE hr.employees  
SET UNUSED COLUMN comments CASCADE  
CONSTRAINTS;
```

- **Drop unused columns:**

```
ALTER TABLE hr.employees  
DROP UNUSED COLUMNS CHECKPOINT 1000;
```

- **Continue to drop column operation:**

```
ALTER TABLE hr.employees  
DROP COLUMNS CONTINUE CHECKPOINT 1000;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Using the UNUSED Option

Instead of removing a column from a table, the column can be marked as unused and then removed later. This has the advantage of being relatively quick, because it does not reclaim the disk space because the data is not removed. Columns that are marked as unused can be removed at a later time from the table when there is less activity on the system.

Unused columns act as if they are not part of the table. Queries cannot see data from unused columns. In addition, the names and data types of those columns are not displayed when a DESCRIBE command is executed. A user can add a new column with the same name as an unused column.

An example of setting a column to unused before dropping it is when you want to drop two columns in the same table. When you drop two columns, all rows in the table are updated twice. But, when you set the columns to unused and then drop the columns, the rows are updated only once.

Using the UNUSED Option (continued)

Identifying Tables with Unused Columns

To identify tables with unused columns, you can query the view `DBA_UNUSED_COL_TABS`. It obtains the names of tables that have unused columns and the number of columns that are marked unused. The following query shows that the table `EMPLOYEES` owned by `HR` has one unused column:

```
SQL > SELECT * FROM dba_unused_col_tabs;
OWNER  TABLE_NAME  COUNT
-----
HR      EMPLOYEES      1
```

To identify tables that have partially completed `DROP COLUMN` operations the `DBA_PARTIAL_DROP_TABS` view can be queried.

```
SQL > SELECT * FROM dba_partial_drop_tabs;
OWNER  TABLE_NAME
-----
no rows selected
```

Restrictions on Dropping a Column

You cannot do the following:

- Drop a column from an object type table
- Drop columns from nested tables
- Drop all columns in a table
- Drop a partitioning key column
- Drop a column from tables owned by `SYS`
- Drop a column from an index-organized table if the column is a primary key
- A `LONG` or `LONG RAW` column that is unused but not dropped will prevent an the ability to add a `LONG` or `LONG RAW` column to the table. (This is true even though a `DESCRIBE` of the table appears to show no `LONG` or `LONG RAW` column.)

Obtaining Table Information

Information about tables can be obtained by querying the following views:

- DBA_TABLES
- DBA_OBJECTS

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Obtaining Table Information

Information about tables can be obtained from the data dictionary. To obtain the data object number and the location of the table header for all tables owned by HR, use the following query:

```
SQL> SELECT table_name
       2  FROM dba_tables WHERE owner = 'HR';
TABLE_NAME
-----
COUNTRIES
DEPARTMENTS
DEPARTMENTS_HIST
EMPLOYEES
EMPLOYEES_HIST
JOBS
JOB_HISTORY
LOCATIONS
REGIONS
```

Obtaining Table Information (continued)

```
SQL> SELECT object_name, created
2  FROM DBA_OBJECTS
3  WHERE object_name like 'EMPLOYEES'
4  AND owner = 'HR';
```

OBJECT_NAME	CREATED
EMPLOYEES	16-APR-01

Summary

In this lesson, you should have learned how to:

- **Distinguish between an extended versus a restricted ROWID**
- **Outline the structure of a row**
- **Create regular and temporary tables**
- **Manage storage structures within a table**
- **Reorganize, truncate, and drop a table**
- **Drop a column within a table**
- **Obtaining table information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 11 Overview

This practice covers the following topics:

- **Creating a table**
- **Viewing, marking as unused, and dropping columns within a table**
- **Allocating extents manually**
- **Truncating a table**
- **Obtaining table information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 11 Overview

Note: This practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 11: Managing Tables

- 1 Create the following tables as user SYSTEM for an order entry system that you are implementing now. The tables and the columns are shown below.

Note: When using OEM be sure to set DATE_OF_DELY to NULL.

In addition, you have been informed that in the table ORDERS, rows will be inserted without a value for DATE_OF_DELY, and it will be updated when the order is fulfilled. Use tablespace USERS. You can use the default storage settings.

Table	Column	Data type and size
CUSTOMERS	CUST_CODE	VARCHAR2 (3)
	NAME	VARCHAR2 (50)
	REGION	VARCHAR2 (5)
ORDERS	ORD_ID	NUMBER (3)
	ORD_DATE	DATE
	CUST_CODE	VARCHAR2 (3)
	DATE_OF_DELY	DATE

- 2 Run the lab11_02.sql script to insert rows into the tables.
- 3 Find which files and blocks contain the rows for the ORDERS table.
Hint: Query the DBA_EXTENTS data dictionary view.
- 4 Check the number of extents used by the ORDERS table.
- 5 Allocate an extent manually, with default size, for the ORDERS table and confirm that the extent has been added as specified.
- 6 Create another table, ORDERS2 as a copy of the ORDERS table in the USERS tablespace, with MINEXTENTS equal to 10. Verify that the table has been created with the specified number of extents.
- 7 Truncate the ORDERS table without releasing space and check the number of extents to verify that extents have not been deallocated.
- 8 Truncate the ORDERS2 table, releasing space. How many extents does the table have now?
- 9 Run the lab11_09.sql script to insert some rows into the ORDERS2 table.
- 10 View the columns for the ORDERS2 table. Then mark the DATE_OF_DELY column as UNUSED. View the columns for the ORDERS2 table again. What happens?
- 11 Drop the unused column DATE_OF_DELY.
- 12 Drop the ORDERS2 table.

12

Managing Indexes

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **List the different types of indexes and their uses**
- **Create various types of indexes**
- **Reorganize indexes**
- **Maintain indexes**
- **Monitor the usage of an index**
- **Obtain index information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Classification of Indexes

- **Logical:**
 - **Single column or concatenated**
 - **Unique or nonunique**
 - **Function-based**
 - **Domain**
- **Physical:**
 - **Partitioned or nonpartitioned**
 - **B-tree: Normal or reverse key**
 - **Bitmap**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Classification of Indexes

An index is a tree structure that allows direct access to a row in a table. Indexes can be classified based on their logical design or on their physical implementation. The logical classification groups indexes from an application perspective, while the physical classification is derived from the way the indexes are stored.

Single column and concatenated indexes

A *single column index* has only one column in the index key—for example, an index on the `employees` number column of an `employees` table.

A *concatenated index*, also known as a composite index, is created on multiple columns in a table. Columns in a concatenated index do not need to be in the same order as the columns in the table, nor do they need to be adjacent—for example, an index on the department and job columns of an employee table.

The maximum number of columns in a composite key index is 32. However, the combined size of all the columns cannot exceed roughly one-half (minus some overhead) of the available data space in a data block.

Classification of Indexes (continued)

Unique and nonunique indexes

Indexes can be unique or nonunique. Unique indexes guarantee that no two rows of a table have duplicate values in the key column (or columns). Nonunique indexes do not impose this restriction on the column values.

Function-based indexes

A function-based index is created when using functions or expressions that involve one or more columns in the table that is being indexed. A function-based index precomputes the value of the function or expression and stores it in the index. Function-based indexes can be created as either a B-tree or a bitmap index.

Domain indexes

A domain index is an application-specific (Text, Spatial) index that is created, managed, and accessed by routines supplied by an index type. It is called a domain index because it indexes data in application-specific domains.

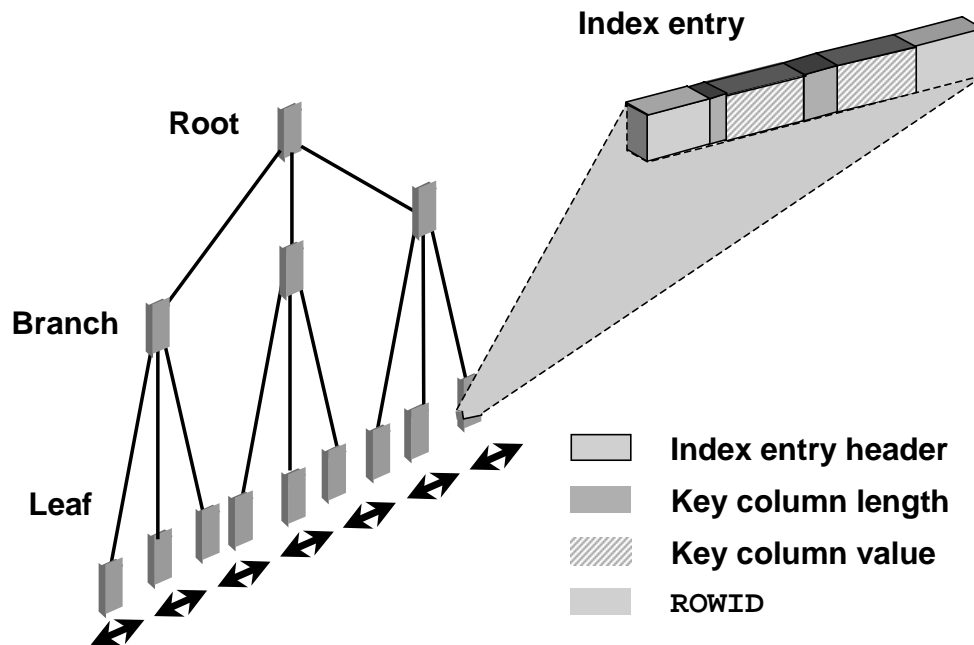
Only single-column domain indexes are supported. You can build single-column domain indexes on columns having scalar, object, or LOB data types.

Partitioned and nonpartitioned indexes

Partitioned indexes are used for large tables to store index entries corresponding to an index in several segments. Partitioning allows an index to be spread across many tablespaces, decreasing contention for index lookup, and increasing manageability. Partitioned indexes are often used with partitioned tables to improve scalability and manageability. An index partition can be created for each table partition.

This lesson discusses the creation and maintenance of nonpartitioned B-tree and bitmap indexes.

B-Tree Index



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

B-Tree Index

Although all the indexes use a B-tree structure, the term B-tree index is usually associated with an index that stores a list of ROWIDs for each key.

Structure of a B-tree index

At the top of the index is the root, which contains entries that point to the next level in the index. At the next level are branch blocks, which in turn point to blocks at the next level in the index. At the lowest level are the leaf nodes, which contain the index entries that point to rows in the table. The leaf blocks are doubly linked to facilitate the scanning of the index in an ascending as well as descending order of key values.

Format of index leaf entries

An index entry is made up of the following components:

- An entry header, which stores the number of columns and locking information
- Key column length-value pairs, which define the size of a column in the key followed by the value for the column (The number of such pairs is a maximum of the number of columns in the index.)
- ROWID of a row, which contains the key values

B-Tree Index (continued)

Index leaf entry characteristics

In a B-tree index on a nonpartitioned table:

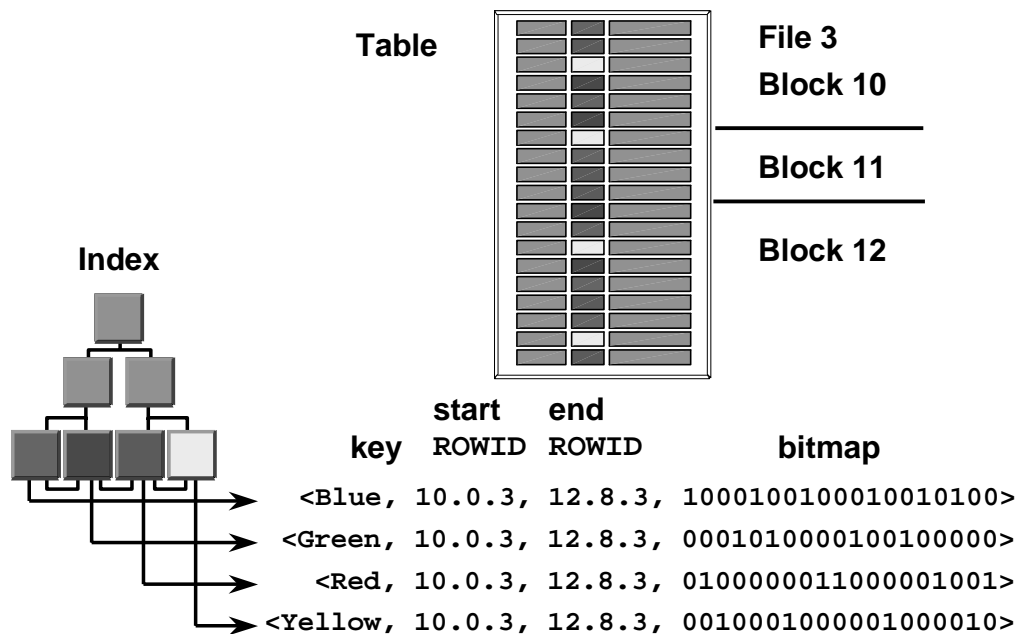
- Key values are repeated if there are multiple rows that have the same key value unless the index is compressed.
- There is no index entry corresponding to a row that has all key columns that are NULL. Therefore, a WHERE clause specifying NULL will always result in a full table scan.
- Restricted ROWID is used to point to the rows of the table because all rows belong to the same segment.

Effect of DML operations on an index

The Oracle server maintains all the indexes when DML operations are carried out on the table. Here is an explanation of the effect of a DML command on an index:

- Insert operations result in the insertion of an index entry in the appropriate block.
- Deleting a row results only in a logical deletion of the index entry. The space used by the deleted row is not available for new entries until all the entries in the block are deleted.
- Updates to the key columns result in a logical delete and an insert to the index. The PCTFREE setting has no effect on the index except at the time of creation. A new entry may be added to an index block even if it has less space than that specified by PCTFREE.

Bitmap Indexes



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Bitmap Indexes

Bitmap indexes are more advantageous than B-tree indexes in certain situations:

- When a table has millions of rows and the key columns have low cardinality—that is, there are very few distinct values for the column. For example, bitmap indexes might be preferable to B-tree indexes for the gender and marital status columns of a table containing passport records.
- When queries often use a combination of multiple WHERE conditions involving the OR operator
- When there is read-only or low update activity on the key columns

Structure of a bitmap index

A bitmap index is also organized as a B-tree, but the leaf node stores a bitmap for each key value instead of a list of ROWIDs. Each bit in the bitmap corresponds to a possible ROWID, and if the bit is set, it means that the row with the corresponding ROWID contains the key value.

As shown in the diagram, the leaf node of a bitmap index contains the following:

- An entry header that contains the number of columns and lock information
- Key values consisting of length and value pairs for each key column. In the example, the key consists of only one column, and the first entry has a key value of Blue.

Bitmap Indexes (continued)

Structure of a bitmap index (continued)

- Start ROWID, which in the example contains a file number three, a block number ten, and a row number zero
- End ROWID, which in the example includes a block number twelve and a row number eight
- A bitmap segment consisting of a string of bits. (The bit is set when the corresponding row contains the key value and is unset when the row does not contain the key value. The Oracle server uses a patented compression technique to store bitmap segments.)

The start ROWID is the ROWID of the first row pointed to by the bitmap segment of the bitmap—that is, the first bit of the bitmap corresponds to that ROWID, the second bit of the bitmap corresponds to the next row in the block, and the end ROWID is a pointer to the last row in the table covered by the bitmap segment. Bitmap indexes use restricted ROWIDs.

Using a bitmap index

The B-tree is used to locate the leaf nodes that contain bitmap segments for a given value of the key. Start ROWID and the bitmap segments are used to locate the rows that contain the key value.

When changes are made to the key column in the table, bitmaps must be modified. This results in locking of the relevant bitmap segments. Because locks are acquired on the whole bitmap segment, a row that is covered by the bitmap cannot be updated by other transactions until the first transaction ends.

Comparing B-Tree and Bitmap Indexes

B-tree	Bitmap
Suitable for high-cardinality columns	Suitable for low-cardinality columns
Updates on keys relatively inexpensive	Updates to key columns very expensive
Inefficient for queries using OR predicates	Efficient for queries using OR predicates
Useful for OLTP	Useful for data warehousing

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Comparing B-Tree and Bitmap Indexes

Bitmap indexes are more compact than B-tree indexes when used with low-cardinality columns.

Updates to key columns in a bitmap index are more expensive because bitmaps use bitmap-segment-level locking, whereas in a B-tree index, locks are on entries corresponding to individual rows of the table.

Bitmap indexes can be used to perform operations such as bitmap Boolean. The Oracle server can use two bitmap segments to perform a bitwise Boolean and get a resulting bitmap. This allows efficient use of bitmaps in queries that use the Boolean predicate.

In summary, B-tree indexes may be more suitable in an OLTP environment for indexing dynamic tables, whereas bitmap indexes may be useful in data warehouse environments where complex queries are used on large, static tables.

Creating B-Tree Indexes

```
CREATE INDEX hr.employees_last_name_idx  
ON hr.employees(last_name)  
PCTFREE 30  
STORAGE(INITIAL 200K NEXT 200K  
PCTINCREASE 0 MAXEXTENTS 50)  
TABLESPACE indx;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating B-Tree Indexes

An index can be created either in the account of the user who owns the table or in a different account, although it is generally created in the same account as the table.

The statement above creates an index on the EMPLOYEES table using the LAST_NAME column.

Creating B-Tree Indexes (continued)

Syntax Options

UNIQUE: Used to specify a unique index. (Nonunique is the default.)

Schema: Owner of the index or table

Index: Name of the index

Table: Name of the table

Column: Name of the column

ASC/DESC: Indicates whether the index should be created in ascending or descending order

TABLESPACE: Identifies the tablespace where the index will be created

PCTFREE: Amount of space reserved in each block (in percentage of total space minus the block header) at the time of creation for accommodating new index entries

INITRANS: Specifies the number of transaction entries preallocated in each block. (The default and the minimum value is two.)

MAXTRANS: Limits the number of transaction entries that can be allocated to each block. (The default is 255.)

STORAGE clause: Identifies the storage clause that determines how extents are allocated to the index

LOGGING: Specifies that the creation of the index and subsequent operations on the index are logged in the online redo log file. (This is the default.)

NOLOGGING: Specifies that the creation and certain types of data loads are not logged in to the online redo log file

NOSORT: Specifies that the rows are stored in the database in ascending order. Therefore, the Oracle server need not sort the rows while creating the index.

Note

- If **MINIMUM EXTENT** has been defined for the tablespace, the extent sizes for the index are rounded up to the next higher multiple of the **MINIMUM EXTENT** value.
- If the **[NO] LOGGING** clause is omitted, the logging attribute of the index defaults to the logging attribute of the tablespace in which it resides.
- **PCTUSED** cannot be specified for an index. Because index entries must be stored in the correct order, the user cannot control when an index block is used for inserts.
- If the **NOSORT** keyword is used when the data is not sorted on the key, the statement terminates with an error. This option is likely to fail if the table has had several DML operations on it.
- The Oracle server uses existing indexes to create a new index, if possible. This happens when the key for the new index corresponds to the leading part of the key of an existing index.

Creating B-Tree Indexes (continued)

Using Oracle Enterprise Manager to Create an Index

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Indexes.
2. Select Create from the right mouse menu.
3. Enter appropriate information in the General, Partitions, Storage, and Options tabbed pages.
4. Click the Create button.

Create Index - sys@DBA01_STC-SUN01.US.ORACLE.COM

General Partitions Storage Options

Name: EMPLOYEES_LAST_NAME_IDX

Schema: HR

Tablespace: INDX

Index On: ☒ Table ☐ Cluster

Schema: HR

Table: EMPLOYEES

Table Columns	Datatype	Order
COMMISSION_PCT	NUMBER	
DEPARTMENT_ID	NUMBER	
EMAIL	VARCHAR2	
EMPLOYEE_ID	NUMBER	
FIRST_NAME	VARCHAR2	
HIRE_DATE	DATE	
JOB_ID	VARCHAR2	
LAST_NAME	VARCHAR2	

Options

☐ Unique ☐ Bitmap ☐ Unsorted ☐ Reverse

Create Cancel Show SQL Help

Creating Indexes: Guidelines

- **Balance query and DML needs.**
- **Place in separate tablespace.**
- **Use uniform extent sizes: Multiples of five blocks or `MINIMUM EXTENT` size for tablespace.**
- **Consider `NOLOGGING` for large indexes.**
- **`INITTRANS` should generally be higher on indexes than on the corresponding tables.**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating Indexes: Guidelines

Consider the following while creating an index:

- Indexes speed up query performance and slow down DML operations. Always minimize the number of indexes needed on volatile tables.
- Place indexes in a separate tablespace, not in a tablespace that has undo segments, temporary segments, or tables.
- There could be significant performance gain for large indexes by avoiding redo generation. Consider using the `NOLOGGING` clause for creating large indexes.
- Because index entries are smaller compared to the rows they index, index blocks tend to have more entries per block. For this reason, `INITTRANS` should generally be higher on indexes than on the corresponding tables.

Indexes and `PCTFREE`

The `PCTFREE` parameter for an index performs differently from that of a table. This parameter is used only during the creation of the index to reserve space for index entries that may need to be inserted into the same index block. Index entries are not updated. When a key column is updated, this involves a logical delete of the index entry and an insert.

Creating Indexes: Guidelines (continued)

Indexes and PCTFREE (continued)

Use a low PCTFREE for indexes on columns that are increasing, such as a system-generated invoice number. In these cases, new index entries are always appended to the existing entries and there is no need to insert a new entry between two existing index entries.

Where the value for an indexed column of an inserted row can take on any value, that is, the new value can fall within the current range of values—you should provide for a higher PCTFREE. An example of an index requiring a high PCTFREE is an index on the customer code column of an invoice table. In this case, it is useful to specify a value of PCTFREE as indicated by the following equation:

$$\frac{(\text{Maximum number of rows} - \text{Initial number of rows} \times 100)}{\text{Maximum number of rows}}$$

The maximum value can cater to a specific time period, such as a year.

Creating Bitmap Indexes

```
CREATE BITMAP INDEX orders_region_id_idx
ON orders(region_id)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K
PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE indx;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating Bitmap Indexes

Syntax

Use the following command to create a bitmap index:

```
CREATE BITMAP INDEX [schema.] index
ON [schema.] table
(column [ ASC | DESC ] [ , column [ASC | DESC ] ] ...)
[ TABLESPACE tablespace ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ storage-clause ]
[ LOGGING | NOLOGGING ]
[ NOSORT ]
```

Notice that a bitmap index cannot be unique.

Creating Bitmap Indexes (continued)

CREATE_BITMAP_AREA_SIZE parameter

The `CREATE_BITMAP_AREA_SIZE` initialization parameter determines the amount of space that will be used for storing bitmap segments in memory. The default value is 8 MB. A larger value may lead to a faster index creation. If cardinality is very small, this value can be set to a small value. For example, if cardinality is only two, then the value can be in the order of kilobytes rather than megabytes. As a general rule, for a higher cardinality, more memory is needed for optimal performance.

Creating Bitmap Indexes (continued)

Using Oracle Enterprise Manager to Create a Bitmap Index

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Indexes.
2. Select Create from the right mouse menu.
3. Enter appropriate information in the General, Partitions, Storage, and Options tabbed pages.
4. Select the Bitmap check box from the General tabbed page.
5. Click the Create button.

Create Index - sys@DBA01_STC-SUN01.US.ORACLE.COM

General Partitions Storage Options

Name: ORDERS_REGION_ID_IDX

Schema: OE

Tablespace: INDX

Index On: ☒ Table ☐ Cluster

Schema: OE

Table: ORDERS

Table Columns	Datatype	Order
CUSTOMER_ID	NUMBER	
ORDER_DATE	TIMESTAMP(6) WITH LO...	
ORDER_ID	NUMBER	
ORDER_MODE	VARCHAR2	
ORDER_STATUS	NUMBER	
ORDER_TOTAL	NUMBER	
PROMOTION_ID	NUMBER	
SALES_REP_ID	NUMBER	

Options

☐ Unique ☒ Bitmap ☐ Unsorted ☐ Reverse

Create Cancel Show SQL Help

Changing Storage Parameters for Indexes

```
ALTER INDEX employees_last_name_idx  
STORAGE(NEXT 400K  
MAXEXTENTS 100);
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Changing Storage Parameters for Indexes

Some of the storage parameters and block utilization parameters can be modified by using the `ALTER INDEX` command.

Syntax

```
ALTER INDEX [schema.]index  
[ storage-clause ]  
[ INITRANS integer ]  
[ MAXTRANS integer ]
```

The implications of changing the storage parameters for an index are the same as the implications of changing them for a table. A common use of this change is to increase the `MAXEXTENTS` for an index.

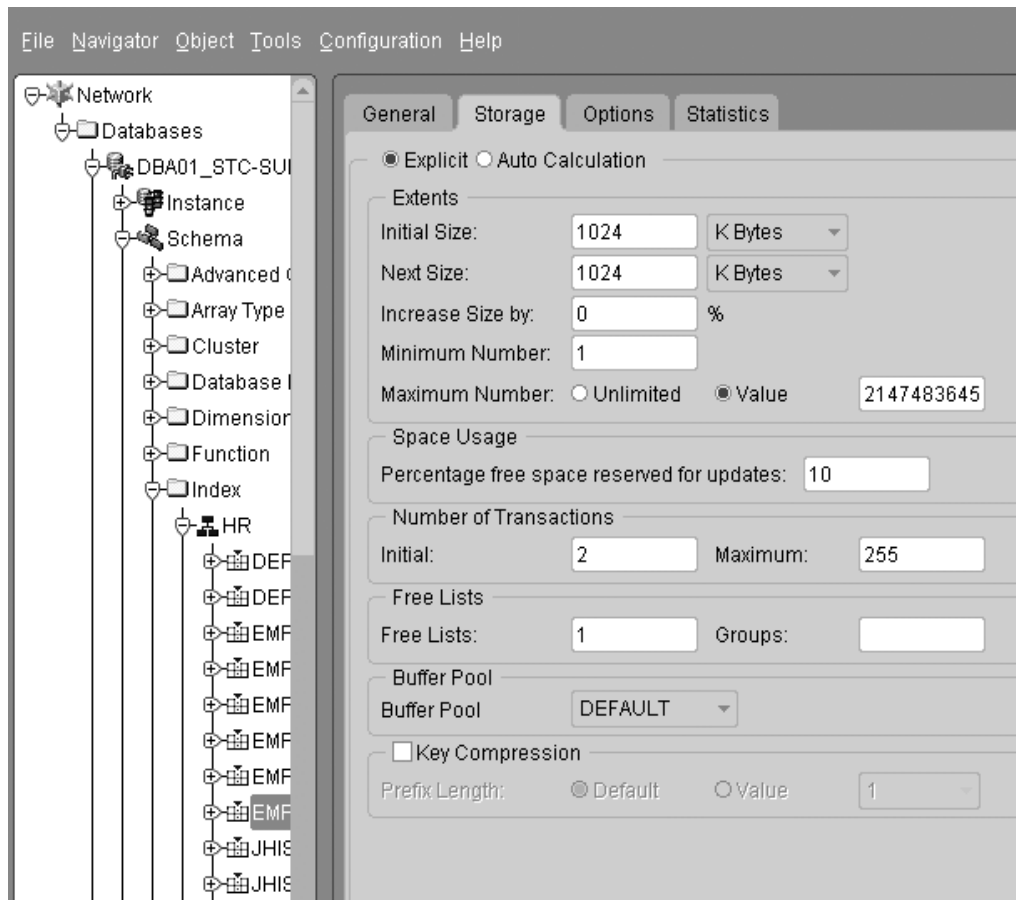
Block utilization parameters can be changed to guarantee higher levels of concurrency on an index block.

Changing Storage Parameters for Indexes (continued)

Using Oracle Enterprise Manager to Change the Storage Parameters of an Index

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Indexes.
2. Highlight the index.
3. Modify the values in the Storage tabbed page.
4. Click Apply.



Allocating and Deallocating Index Space

```
ALTER INDEX orders_region_id_idx  
ALLOCATE EXTENT (SIZE 200K  
DATAFILE '/DISK6/indx01.dbf');
```

```
ALTER INDEX orders_id_idx  
DEALLOCATE UNUSED;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Allocating and Deallocating Index Space

Manual allocation of space to an index

It may be necessary to add extents to an index before a period of high insert activity on a table. Adding extents prevents dynamic extension of indexes and the resulting degradation in performance.

Manual deallocation of space from an index

Use the DEALLOCATE clause of the ALTER INDEX command to release unused space above the high-water mark in an index.

Syntax

Use the following command to allocate or deallocate index space:

```
ALTER INDEX [schema.]index  
{ALLOCATE EXTENT ([SIZE integer [K|M]]  
  [ DATAFILE 'filename' ] )  
| DEALLOCATE UNUSED [KEEP integer [ K|M ] ] }
```

Manual allocation and deallocation of space for an index follow the same rules as those that are used when using these commands against a table.

Note: Index space is deallocated when the table is truncated. Truncating a table results in truncation of the associated index.

Rebuilding Indexes

Use the **ALTER INDEX** command to:

- **Move an index to a different tablespace**
- **Improve space utilization by removing deleted entries**

```
ALTER INDEX orders_region_id_idx REBUILD  
TABLESPACE indx02;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Rebuilding Indexes

Index rebuilds have the following characteristics:

- A new index is built using an existing index as the data source.
- Sorts are not needed when an index is built using an existing index, resulting in better performance.
- The old index is deleted after the new index is built. During the rebuild, sufficient space is needed to accommodate both the old and the new index in their respective tablespaces.
- The resulting index does not contain any deleted entries. Therefore, this index uses space more efficiently.
- Queries can continue to use the existing index while the new index is being built.

Possible rebuild situations

Rebuild an index in the following situations:

- The existing index must be moved to a different tablespace. This may be necessary if the index is in the same tablespace as the table or if objects need to be redistributed across disks.

Rebuilding Indexes (continued)

Possible rebuild situations (continued):

- An index contains many deleted entries. This is a typical problem with sliding indexes, such as an index on the order number of an orders table, where completed orders are deleted and new orders with higher numbers are added to the table. If a few old orders are outstanding, there may be several index leaf blocks with all but a few deleted entries.
- An existing normal index must be converted into a reverse key index. This may be the case when migrating applications from an earlier release of the Oracle server.
- The table of the index has been moved to another tablespace using the `ALTER TABLE ... MOVE TABLESPACE` command.

Syntax

Use the following command to rebuild an index:

```
ALTER INDEX [schema.] index REBUILD
[ TABLESPACE tablespace ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ storage-clause ]
[ LOGGING | NOLOGGING ]
[ REVERSE | NOREVERSE ]
```

The `ALTER INDEX ... REBUILD` command cannot be used to change a bitmap index to B-tree and vice versa. The `REVERSE` or `NOREVERSE` keyword can be specified only for B-tree indexes.

Rebuilding Indexes Online

- Indexes can be rebuilt with minimal table locking.

```
ALTER INDEX orders_id_idx REBUILD ONLINE;
```

- Some restrictions still apply.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Rebuilding Indexes Online

Building or rebuilding an index can be a time-consuming task, especially if the table is very large. Before Oracle8i, creating or rebuilding indexes required a lock on the table and prevented concurrent DML operations.

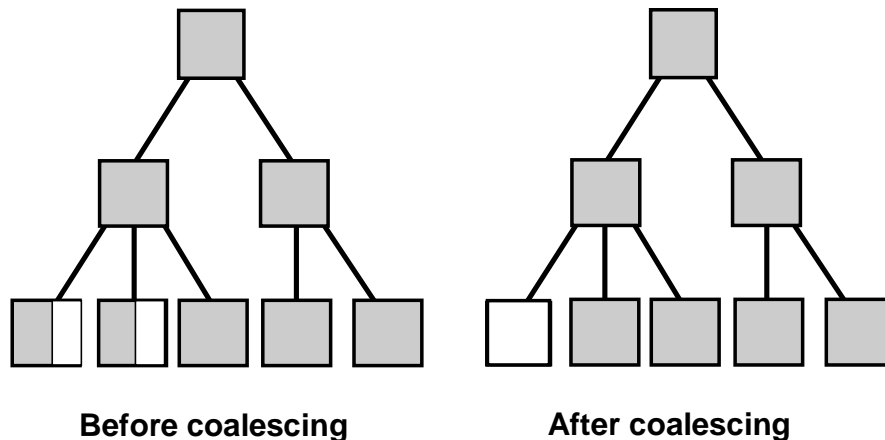
Oracle9i offers a method of creating or re-creating an index while allowing concurrent operations on the base table, but performing large DML operations during this procedure is not recommended.

Note: DML locks are still there, which means you cannot perform other DDL operations during an online index build.

Restrictions

- You cannot rebuild an index on a temporary table.
- You cannot rebuild an entire partitioned index. You must rebuild each partition or subpartition.
- You cannot change the value of the PCTFREE parameter for the index as a whole.

Coalescing Indexes



```
ALTER INDEX orders_id_idx COALESCE;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Coalescing Indexes

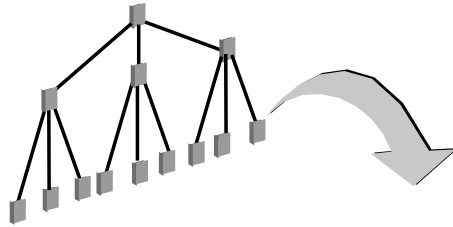
When you encounter index fragmentation, you can rebuild or coalesce the index. Before you perform either task, you should consider the cost and benefits of each option and choose the one that works best for your situation. Coalesce on an index is a block rebuild that is performed online.

In situations where you have B-tree index leaf blocks that can be freed for reuse, you can merge those leaf blocks using the following SQL statement:

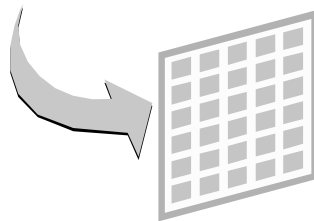
```
SQL> ALTER INDEX hr.employees_idx COALESCE;
```

The figure above shows the effect of `ALTER INDEX ... COALESCE` on the `hr.employees_idx` index. Before performing the COALESCE operation, the first two leaf blocks are 50% full. This means the index is fragmented and can be coalesced to completely fill the first block, reducing fragmentation.

Checking Index Validity



```
ANALYZE INDEX orders_region_id_idx  
VALIDATE STRUCTURE;
```



INDEX_STATS

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Checking Index Validity

Analyze the index to perform the following:

- Check all the index blocks for block corruption. Note that this command does not verify whether index entries correspond to data in the table.
- Populate the INDEX_STATS view with information about the index.

Syntax

```
ANALYZE INDEX [ schema.]index VALIDATE STRUCTURE
```

After running this command, query INDEX_STATS to obtain information about the index as shown in the following example:

Checking Index Validity (continued)

```
SQL> SELECT blocks, pct_used, distinct_keys  
2 lf_rows, del_lf_rows  
3 FROM index_stats;
```

BLOCKS	PCT_USED	LF_ROWS	DEL_LF_ROWS
25	11	14	0

1 row selected.

Reorganize the index if it has a high proportion of deleted rows, for example, when the ratio of DEL_LF_ROWS to LF_ROWS exceeds 30%.

Dropping Indexes

- Drop and re-create an index before bulk loads.
- Drop indexes that are infrequently needed, and build indexes when necessary.
- Drop and re-create invalid indexes.

```
DROP INDEX hr.departments_name_idx;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dropping Indexes

Indexes may need to be dropped in the following scenarios:

- An index that is no longer needed by applications can be removed.
- An index might be dropped prior to performing bulk loads. Dropping an index prior to large data loads and re-creating them after the load:
 - Improves performance of the load
 - Uses index space more efficiently
- Indexes that are used only periodically do not need to be maintained unnecessarily, especially if they are based on volatile tables. This is generally the case in an OLTP system, where ad hoc queries are generated at year end or quarter end to gather information for review meetings.
- An index might be marked `INVALID` when there is an instance failure during certain types of operations such as loading. In this case, the index must be dropped and re-created.
- The index is corrupt.

Because indexes that are required for constraints cannot be dropped, the dependent constraint must be disabled or dropped first.

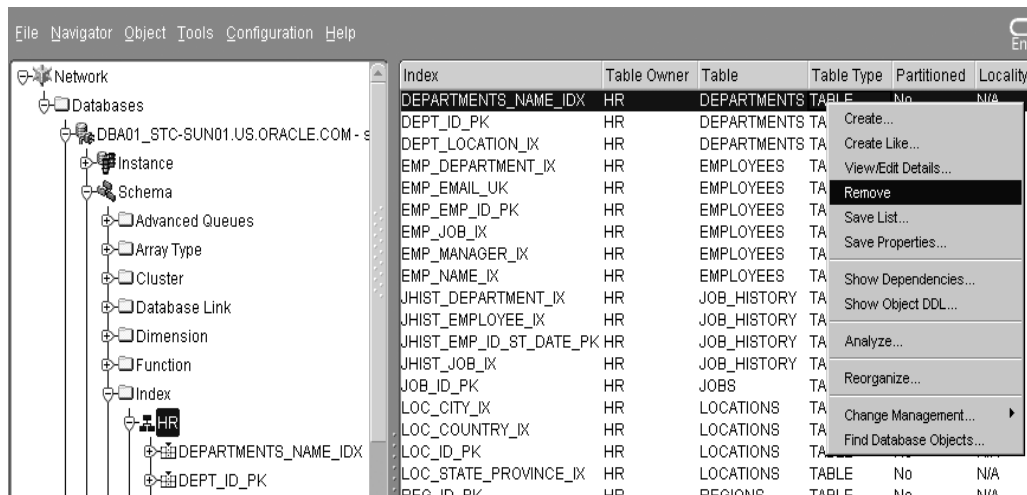
Dropping Indexes (continued)

Using Oracle Enterprise Manager to Drop an Index

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Indexes.
2. Highlight the index.
3. Select Remove from right mouse menu.
4. Click the Yes button to confirm the drop.

Note: An index cannot be dropped if it is used to implement an integrity constraint that is enabled. Constraints are discussed in the “Maintaining Data Integrity” lesson.



Identifying Unused Indexes

- To start monitoring the usage of an index:

```
ALTER INDEX hr.dept_id_idx  
MONITORING USAGE
```

- To stop monitoring the usage of an index:

```
ALTER INDEX hr.dept_id_idx  
NOMONITORING USAGE
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Identifying Unused Indexes

Beginning with Oracle9i, statistics about the usage of an index can be gathered and displayed in V\$OBJECT_USAGE. If the information gathered indicates that an index is never used, the index can be dropped. In addition, eliminating unused indexes reduces the overhead required of the Oracle server for DML, thus improving performance. Each time the MONITORING USAGE clause is specified, V\$OBJECT_USAGE will be reset for the specified index. The previous information is cleared or reset, and a new start time is recorded.

V\$OBJECT_USAGE columns

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

Obtaining Index Information

Information about indexes can be obtained by querying the following views:

- **DBA_INDEXES:** Provides information on the indexes
- **DBA_IND_COLUMNS:** Provides information on the columns indexed
- **V\$OBJECT_USAGE:** Provides information on the usage of an index

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Summary

In this lesson, you should have learned how to:

- **Create different types of indexes**
- **Reorganize indexes**
- **Drop indexes**
- **Get index information from the data dictionary**
- **Begin and end the monitoring of index usage**
- **Obtaining index information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 12 Overview

This practice covers the following topics:

- **Creating an index on columns of a table**
- **Moving the index to another tablespace**
- **Dropping an index**
- **Obtaining index information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 12 Overview

Note: Practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 12: Solutions

- 1 You are considering creating indexes on the NAME and REGION columns of the CUSTOMERS table. What types of index are appropriate for the two columns? Create two indexes, naming them CUST_NAME_IDX and CUST_REGION_IDX, respectively, and place them in the INDEX01 tablespace.

Hint: A B-tree index is suitable for a column with many distinct values, and a bitmap index is suitable for columns with only a few distinct values.

Note: CUSTOMERS table is in the SYSTEM schema.

- 2 Move the CUST_REGION_IDX index to another tablespace.

Hint: The index can be rebuilt specifying a different tablespace.

- 3 Note the files and blocks used by the extents for the CUST_REGION_IDX index.

Hint: Use the view DBA_EXTENTS to get this information.

Note: The owner is SYSTEM.

- 4 Re-create the CUST_REGION_IDX index without dropping and re-creating it, and retain it in the same tablespace as before. Does the new index use the same blocks that were used earlier?

Hint: Rebuild the index.

Note: The new index does not reuse the same space as seen from the location of the extent after rebuild. This is because Oracle server builds a temporary index, drops the old one, and renames the temporary index.

- 5
 - a As user SYSTEM, run the script lab12_05a.sql to create and populate the NUMBERS table.
 - b Query the NUMBERS table to find the number of distinct values in the two columns in the table.
 - c Using uniform extent sizes of 4KB, create two indexes, NUMB_OE_IDX and NUMB_NO_IDX, on the ODD_EVEN and NO columns of the NUMBERS table, respectively. Place the indexes in the INDEX01 tablespace. Check the total sizes of the indexes and write the number of blocks in the box below.

Hint: Use PCTINCREASE equal to zero to create extents of equal size. Check the total blocks allocated to the extents from DBA_SEGMENTS.

Index	Column	Blocks
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

Practice 12: Solutions (continued)

- 5 d** After you have recorded the blocks above, drop the two indexes, NUMB_OE_IDX and NUMB_NO_IDX . Using uniform extent sizes of 4KB, create bitmap indexes NUMB_OE_IDX and NUMB_NO_IDX on the ODD_EVEN and NO columns of the NUMBERS table, respectively. Place the indexes in the INDEX01 tablespace. Re-execute the query to check the total blocks allocated to the extents from DBA_SEGMENTS. Check the total sizes of the indexes and record in the box below. What can you conclude about the relationship between cardinality and sizes of the two types of indexes?

Index	Column	Blocks
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

13

Maintaining Data Integrity

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

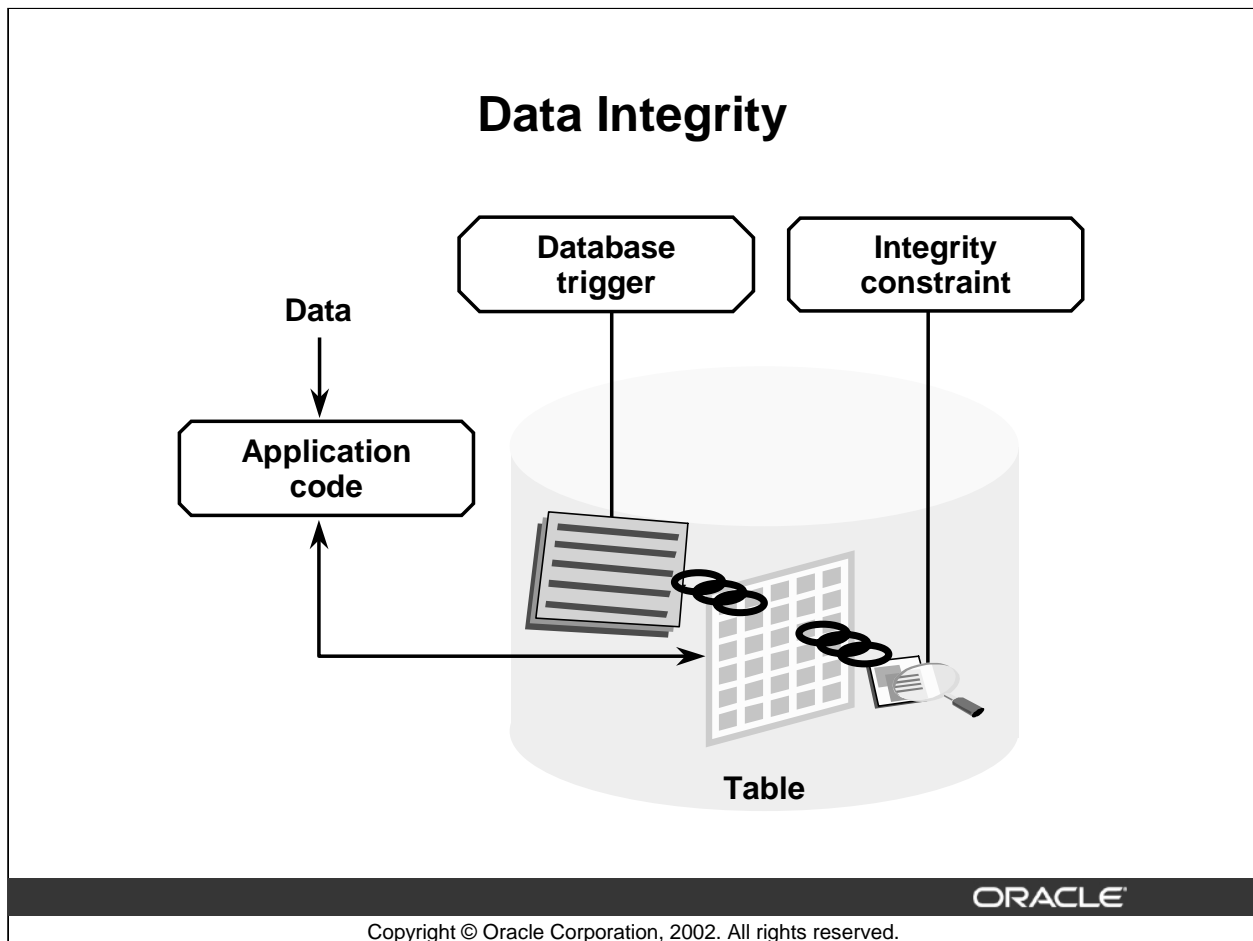
Objectives

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

- **Implement data integrity constraints**
- **Maintain integrity constraints**
- **Obtain constraint information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.



Data Integrity

Data integrity means that data in a database adheres to business rules. There are three primary ways in which data integrity can be maintained:

- Application code
- Database triggers
- Declarative integrity constraints

Mapping the business rules using one of the three methods is a design decision. The database administrator is primarily concerned with implementing the methods chosen by the designer and balancing the performance needs against integrity requirements.

Application code can be implemented either as stored procedures within the database or as applications running on the client. This lesson focuses on the use of integrity constraints.

Data Integrity

Database triggers

Database triggers are PL/SQL programs that are executed when an event, such as an insert or an update of a column, occurs on a table. Triggers can be enabled or disabled—that is, they can be set to execute when the event occurs, or they can be set not to execute even though they are defined. Database triggers are usually created only to enforce a complex business rule that cannot be defined as an integrity constraint.

Note: Database triggers are covered in other Oracle courses.

Integrity constraints

Integrity constraints are the preferred mechanism for enforcing business rules because they:

- Provide improved performance
- Are easy to declare and modify—they do not require extensive coding
- Centralize rules
- Are flexible (enabled or disabled)
- Are fully documented in the data dictionary

The following sections explain the behavior of integrity constraints and discuss how they are implemented by the Oracle server.

Types of Constraints

Constraint	Description
NOT NULL	Specifies that a column cannot contain null values
UNIQUE	Designates a column or combination of columns as unique
PRIMARY KEY	Designates a column or combination of columns as the table's primary key
FOREIGN KEY	Designates a column or combination of columns as the foreign key in a referential integrity constraint
CHECK	Specifies a condition that each row of the table must satisfy

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Types of Constraints

By default, all columns in a table allow nulls. Null means the absence of a value. A NOT NULL constraint requires a table column to contain values.

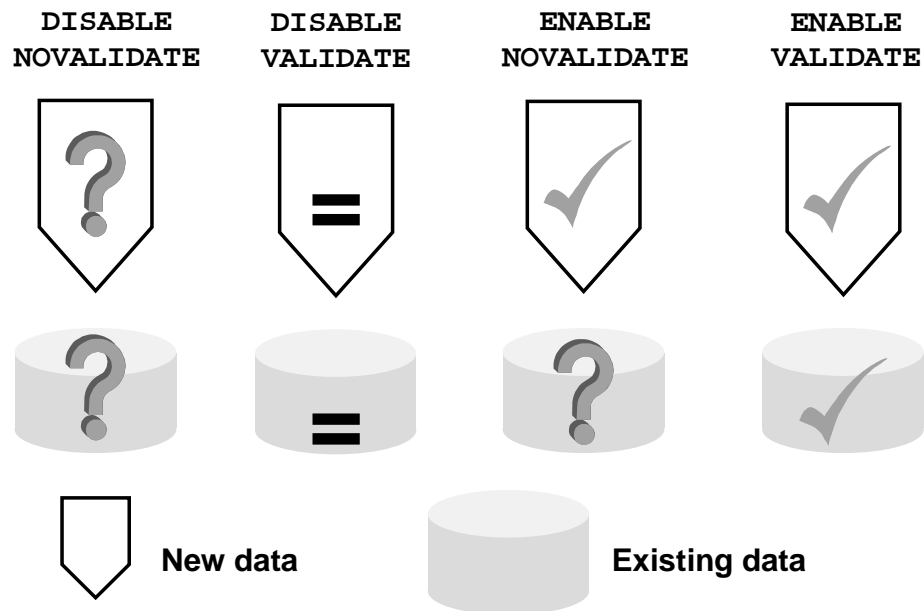
A UNIQUE key constraint requires that every value in a column or set of columns (key) be unique. No two rows in a table can have duplicate values in a specified column or set of columns.

Each table in the database can contain at most one PRIMARY KEY constraint. A PRIMARY KEY constraint ensures that both of the following are true:

- No two rows in a table can have duplicate values in the specified column.
- Primary key columns do not contain NULL values
- A CHECK integrity constraint on a column or a set of columns requires that a specified condition be true or unknown for every row of the table.

Although the NOT NULL and CHECK constraints do not directly require DBA attention, the PRIMARY KEY, UNIQUE, and FOREIGN KEY constraints must be managed to ensure high availability and acceptable performance levels.

Constraint States



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Constraint States

An integrity constraint can be enabled (ENABLE) or disabled (DISABLE). If a constraint is enabled, data is checked as it is entered or updated in the database. Data that does not conform to the constraint's rule is prevented from being entered. If a constraint is disabled, then data that does not conform can be entered into the database. An integrity constraint can be in one of the following states:

- DISABLE NOVALIDATE
- DISABLE VALIDATE
- ENABLE NOVALIDATE
- ENABLE VALIDATE

DISABLE NOVALIDATE: A constraint that is DISABLE NOVALIDATE is not checked. Data in the table, as well as new data that is entered or updated, may not conform to the rules defined by the constraint.

DISABLE VALIDATE: If a constraint is in this state, then any modification of the constrained columns is not allowed. In addition, the index on the constraint is dropped and the constraint is disabled.

Note: The index is not dropped if the constraint is deferrable.

Constraint States (continued)

ENABLE NOVALIDATE: If a constraint is in this state, new data that violates the constraint cannot be entered. However, the table may contain data that is invalid—that is, data that violates the constraint. Enabling constraints in the NOVALIDATE state is most useful in data warehouse configurations that are uploading valid OLTP data.

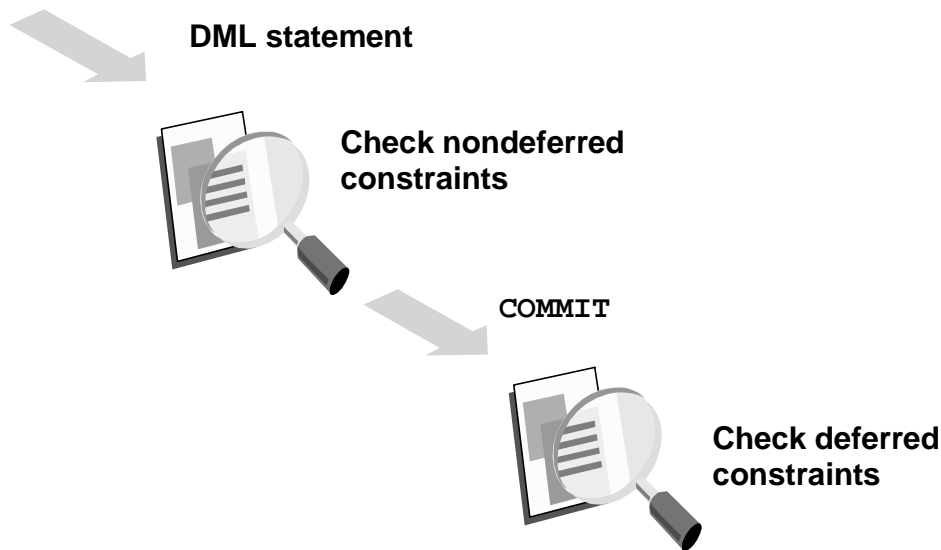
ENABLE VALIDATE: If a constraint is in this state, a row that violates the constraint cannot be inserted into the table. However, such a row can be inserted while the constraint is disabled. This row is known as an exception to the constraint. If the constraint is in the ENABLE NOVALIDATE state, then the violations resulting from data that is entered while the constraint is disabled remain. The rows that violate the constraint must be either updated or deleted in order for the constraint to be put in the validated state.

When a constraint changes to ENABLE VALIDATE from a disabled state, the table is locked and all data in the table is checked for conformity. This may cause DML operations such as a data load to wait, so it is advisable to move first from a disabled state to ENABLE NOVALIDATE, and then to ENABLE VALIDATE.

Transitions between these states are governed by the following rules:

- ENABLE implies VALIDATE, unless NOVALIDATE is specified.
- DISABLE implies NOVALIDATE, unless VALIDATE is specified.
- VALIDATE and NOVALIDATE do not have default implications for the ENABLE and DISABLE states.
- When a unique or primary key moves from the DISABLE state to the ENABLE state and there is no existing index, a unique index is created automatically. (The exception will exist if the index is deferrable.) Similarly, when a unique or primary key moves from ENABLE to DISABLE and it is enabled with a unique index, the unique index is dropped.
- When any constraint is moved from the NOVALIDATE state to the VALIDATE state, all data must be checked. However, moving from VALIDATE to NOVALIDATE, it simply forgets that the data was ever checked.
- Moving a single constraint from the ENABLE NOVALIDATE state to the ENABLE VALIDATE state does not block reads, writes, or other DDL statements.

Constraint Checking



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Constraint Checking

You can defer checking constraints for validity until the end of the transaction.

Nondeferred or immediate constraints

Nondeferred constraints, also known as immediate constraints, are enforced at the end of every DML statement. A constraint violation causes the statement to be rolled back. If a constraint causes an action such as `delete cascade`, then the action is taken as part of the statement that caused it. A constraint that is defined as nondeferrable cannot be modified to be enforced at the end of a transaction.

Deferred constraints

Deferred constraints are constraints that are checked only when a transaction is committed. If any constraint violations are detected at commit time, the entire transaction is rolled back. These constraints are most useful when both the parent and child rows in a foreign key relationship are entered at the same time, as in the case of an order entry system, where the order and the items in the order are entered at the same time.

A constraint that is defined as deferrable can be specified as one of the following:

- *Initially immediate* specifies that by default it should function as an immediate constraint, unless explicitly set otherwise.
- *Initially deferred* specifies that by default the constraint should be enforced only at the end of the transaction.

Defining Constraints Immediate or Deferred

- Use the **SET CONSTRAINTS** statement to make constraints either **DEFERRED** or **IMMEDIATE**.
- The **ALTER SESSION** statement also has clauses to set constraints to **DEFERRED** or **IMMEDIATE**.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Defining Constraints Immediate or Deferred

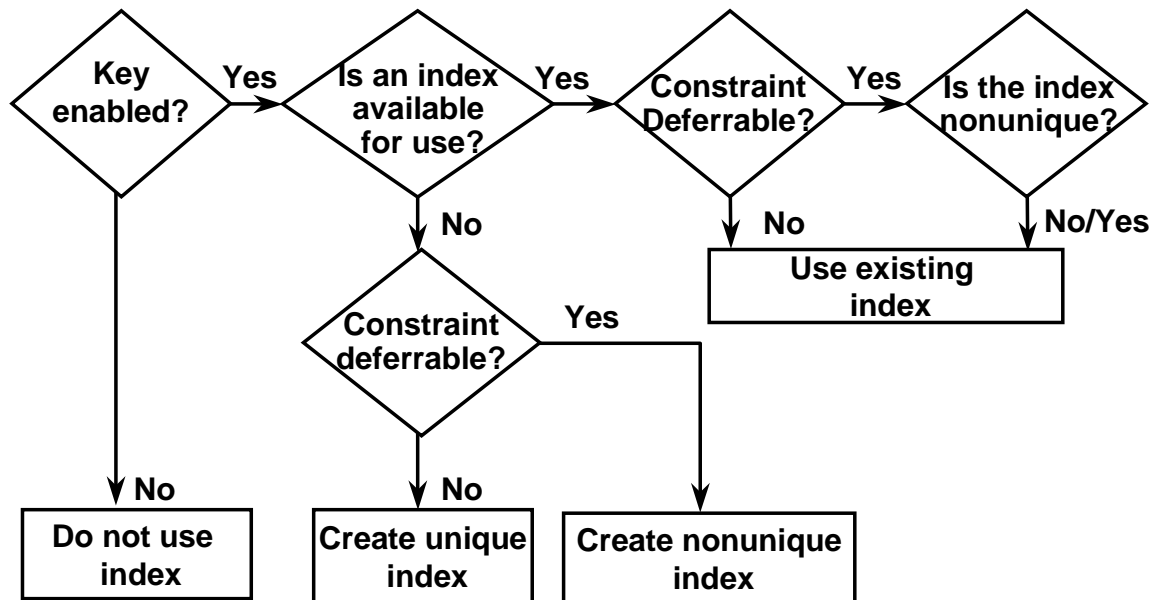
The **SET CONSTRAINTS** statement makes constraints either **DEFERRED** or **IMMEDIATE** for a particular transaction. You can use this statement to set the mode for a list of constraint names or for constraints. The **SET CONSTRAINTS** mode lasts for the duration of the transaction or until another **SET CONSTRAINTS** statement resets the mode. The **SET CONSTRAINTS** statement is disallowed inside triggers.

The **ALTER SESSION** statement also has clauses to set constraints to **IMMEDIATE** or **DEFERRED**. This command implies setting **ALL** deferrable constraints (list of constraint names cannot be specified). The **ALTER SESSION SET CONSTRAINTS** statement applies to a current session only.

```
ALTER SESSION
  SET CONSTRAINT[S] =
    { IMMEDIATE | DEFERRED | DEFAULT }
```

```
SET CONSTRAINT | CONSTRAINTS
  { constraint | ALL }
  { IMMEDIATE | DEFERRED }
```

Primary and Unique Key Enforcement



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Primary and Unique Key Enforcement

Primary and unique keys are enforced using indexes. You can control the location and type of index that is used for enforcing these constraints.

The Oracle server uses the following procedure to implement unique and primary key constraints:

- If the constraint is disabled, no indexes are needed.
- If the constraint is enabled and the columns in the constraint form the leading part of an index, the index is used to enforce the constraint whether the index itself was created as unique or nonunique.
- If the constraint is enabled and there is no index that uses the constraint columns as a leading part of the index, then an index with the same name as the constraint is created using the following rules:
 - If the key is deferrable, a nonunique index on the key column is created.
 - If the key is nondeferrable, a unique index is created.
- If an index is available for use and the constraint is nondeferrable, then use the existing index. If the constraint is deferrable and the index is nonunique, then use the existing index.

Foreign Key Considerations

Desired Action	Appropriate Solution
Drop parent table	Cascade constraints
Truncate parent table	Disable or drop foreign key
Drop tablespace containing parent table	Use the <code>CASCADE CONSTRAINTS</code> clause
Perform DML on child table	Ensure that the tablespace containing the parent key is online

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Foreign Key Considerations

You should consider several factors when you maintain tables that are in a foreign key relationship.

DDL involving parent table

The foreign key must be dropped before dropping the parent table. The following single statement can be used to perform both actions:

```
DROP TABLE table CASCADE CONSTRAINTS
```

The parent table cannot be truncated without dropping or disabling the foreign key.

The foreign key must be dropped before the tablespace containing the parent is dropped.

You can achieve this by using the following command:

```
DROP TABLESPACE tablespace INCLUDING CONTENTS  
CASCADE CONSTRAINTS
```

Foreign Key Considerations (continued)

If the `DELETE CASCADE` option is not used when rows are deleted from the parent table, the Oracle server must ensure that there are no rows in the child table with the corresponding foreign key. Similarly, an update to the parent key is permitted only when there are no child rows with the old key value. If there is no index on the foreign key on the child table, the Oracle server locks the child table and prevents changes to ensure referential integrity. If there is an index on the table, the referential integrity is maintained by locking the index entries and avoiding more restrictive locks on the child table. If both tables must be updated concurrently from different transactions, create an index on the foreign key columns.

When data is inserted into the child table, or the foreign key column is updated in the child table, the Oracle server checks the index on the parent table that is used for enforcing the referenced key. Therefore, the operation succeeds only if the tablespace that contains the index is online. Note that the tablespace that contains the parent table does not need to be online to perform DML operations on the child table.

Oracle9i no longer requires a share lock on unindexed foreign keys when doing an update or delete on the primary key. It still obtains the table-level share lock, but then releases it immediately after obtaining it. If multiple primary keys are updated or deleted, the lock is obtained and released once per row.

Defining Constraints While Creating a Table

```
CREATE TABLE hr.employee(  
  id NUMBER(7)  
    CONSTRAINT employee_id_pk PRIMARY KEY  
    DEFERRABLE  
    USING INDEX  
    STORAGE(INITIAL 100K NEXT 100K)  
    TABLESPACE indx,  
  last_name VARCHAR2(25)  
    CONSTRAINT employee_last_name_nn NOT NULL,  
  dept_id NUMBER(7))  
  TABLESPACE users;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Defining Constraints While Creating a Table

A constraint can be defined either when a table is created or when a table is altered. Use the `constraint_clause` clause in a `CREATE TABLE` or `ALTER TABLE` statement to define a constraint. You must have the requisite privileges to define an integrity constraint. To create a referential integrity constraint, the parent table must be in your own schema, or you must have the `REFERENCES` privilege on the columns of the referenced key in the parent table.

Defining Constraints While Creating a Table (continued)

The `column_constraint` syntax is part of the table definition. At the time the table is created, you can define the constraint by using the following syntax:

```
column datatype
[CONSTRAINT constraint]
{[NOT] NULL
|UNIQUE [USING INDEX index_clause]
|PRIMARY KEY [USING INDEX index_clause]
|REFERENCES [schema.]table [(column)]
|ON DELETE CASCADE}
|CHECK (condition)}
constraint_state ::=
[NOT DEFERRABLE|DEFERRABLE [INITIALLY {IMMEDIATE|DEFERRED}]]
[DISABLE|ENABLE [VALIDATE|NOVALIDATE]]
```

where:

CONSTRAINT: Identifies the integrity constraint by the name `constraint` stored in data dictionary

USING INDEX: Specifies that the parameters defined in the `index_clause` should be used for the index that the Oracle server uses to enforce a unique or primary key constraint (The name of the index is the same as the name of the constraint.)

DEFERRABLE: Indicates that constraint checking can be deferred until the end of the transaction by using the `SET CONSTRAINT` command

NOT DEFERRABLE: Indicates that this constraint is checked at the end of each DML statement (A `NOT DEFERRABLE` constraint cannot be deferred by sessions or transactions. `NOT DEFERRABLE` is the default.)

INITIALLY IMMEDIATE: Indicates that at the start of every transaction, the default is to check this constraint at the end of every DML statement (If no `INITIALLY` clause is specified, `INITIALLY IMMEDIATE` is the default.)

INITIALLY DEFERRED: Implies that this constraint is `DEFERRABLE` and specifies that, by default, the constraint is checked only at the end of each transaction

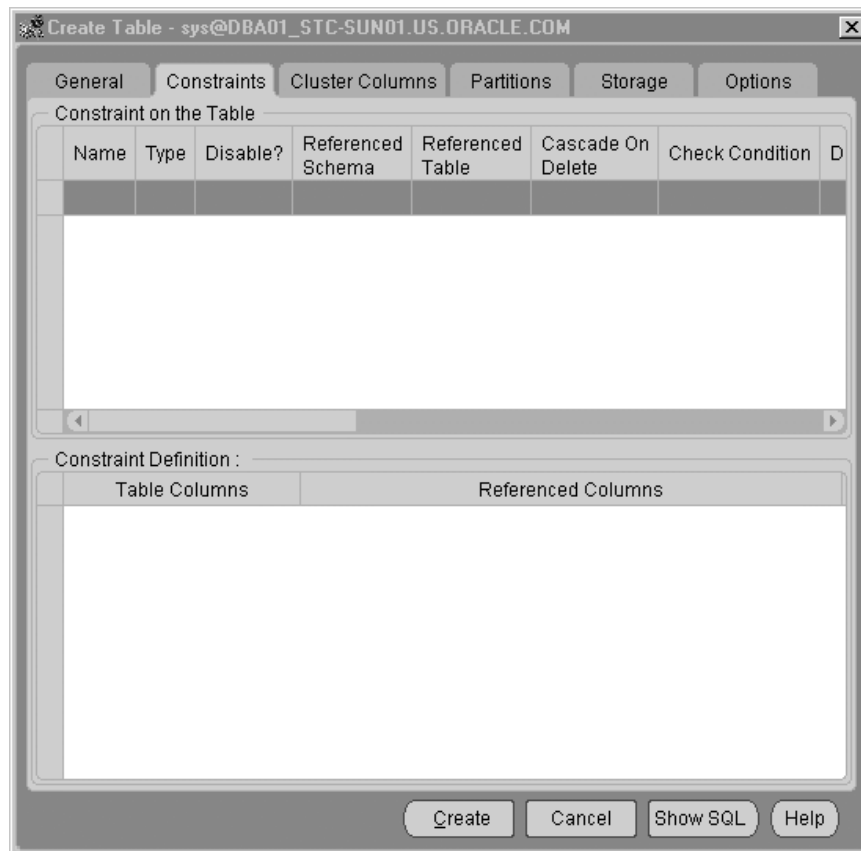
DISABLE: Disables the integrity constraint (If an integrity constraint is disabled, the Oracle server does not enforce it.)

Defining Constraints While Creating a Table (continued)

Using Oracle Enterprise Manager to Define Constraints

From the OEM Console:

1. Navigate to Schema > [Schema name] > Tables.
2. Select Create from the right-mouse menu.
3. Complete appropriate information in the General tabbed page.
4. Open the Constraints tabbed page and define integrity constraints.
5. Click Create .



Defining Constraints While Creating a Table (continued)

Table constraint

A table constraint is part of the table definition. This can define any type of constraint except a NOT NULL constraint. This is defined using the following syntax:

```
column datatype
[CONSTRAINT constraint]
{PRIMARY KEY (column [, column ]... )
[USING INDEX index_clause]
|UNIQUE (column [, column ]... )
[USING INDEX index_clause]
|FOREIGN KEY (column [, column ]... )
REFERENCES [schema.]table [(column [, column ]... )]
[ON DELETE CASCADE]
|CHECK (condition)}
[constraint_state]
```

Note

- It is a good practice to adopt a standard naming convention for constraints. This is especially true with CHECK constraints because the same constraint can be created several times with different names.
- Table constraints are needed in the following cases:
 - When a constraint names two or more columns
 - When a table is altered to add any constraint other than the NOT NULL constraint
- Defining a constraint from the type NOT NULL after creating a table is possible only with:

```
ALTER TABLE table MODIFY column CONSTRAINT constraint NOT NULL;
```

Defining constraints after creating a table: Example

```
SQL> ALTER TABLE hr.employee
2 ADD(CONSTRAINT employee_dept_id_fk FOREIGN KEY(dept_id)
3 REFERENCES hr.department(id)
4 DEFERRABLE INITIALLY DEFERRED);
```

Note: The EXCEPTIONS clause, discussed under “Enabling Constraints” later in this lesson, can be used to identify rows which violate a constraint that is added by using the ALTER TABLE command.

Guidelines for Defining Constraints

- **Primary and unique constraints:**
 - Place indexes in a separate tablespace.
 - Use nonunique indexes if bulk loads are frequent.
- **Self-referencing foreign keys:**
 - Define or enable foreign keys after the initial load.
 - Defer constraint checking.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

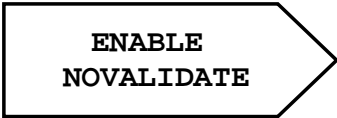
Guidelines for Defining Constraints

The following guidelines are useful when defining constraints:

- Place indexes that are used to enforce primary key and unique constraints in a tablespace different from that of the table. You can do this either by specifying the `USING INDEX` clause or by creating the table, creating the index, and altering the table to add or enable the constraint.
- If data is frequently loaded in bulk into a table, it is preferable to disable the constraints, perform the load, and then enable the constraints. If a unique index is used to enforce a primary key or unique constraint, then this index must be dropped when the constraint is disabled. Performance can be enhanced by using a nonunique index for enforcement of primary key or unique constraints in such situations: either create the key as deferrable or create the index before defining or enabling the key.
- If a table contains a self-referencing foreign key, use one of the following methods to load data:
 - Define or enable the foreign key after the initial load.
 - Define the constraint as a deferrable constraint.

The second method is useful if data loads are performed frequently.

Enabling Constraints



**ENABLE
NOVALIDATE**

- No locks on table
- Primary and unique keys must use nonunique indexes

```
ALTER TABLE hr.departments  
ENABLE NOVALIDATE CONSTRAINT dept_pk;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Enabling Constraints

A constraint that is currently disabled can be enabled in one of the two ways: `ENABLE NOVALIDATE` or `ENABLE VALIDATE`.

Enable **NOVALIDATE**

For `PRIMARY KEY` and `UNIQUE` constraints that have an existing index, enabling a `NOVALIDATE` constraint is much faster than enabling a `VALIDATE` constraint because existing data is not checked for constraint violation if the constraint is deferrable. If this option is used for enabling a constraint, no locks are required on the table. This method is appropriate where there is a lot of DML activity on a table, as in the case of an OLTP environment.

The following command can be used to enable a `ENABLE NOVALIDATE` constraint:

```
ALTER TABLE [ schema. ] table  
ENABLE NOVALIDATE {CONSTRAINT constraint  
| PRIMARY KEY | UNIQUE ( column [, column ] ... ) }  
[ USING INDEX index_clause ]
```

Enabling Constraints (continued)

Restrictions

The USING INDEX clause is applicable only for primary key or unique constraints that were created as deferrable, and when one of the following is true:

- The constraints were created disabled.
- The constraints were disabled and the index was dropped.

However, if the index needs to be created, then using this method of enabling a constraint does not offer any significant benefit over ENABLE VALIDATE because the Oracle server locks the table to build the index.

Note: Disabling constraints is covered in the *Introduction to Oracle9i: SQL Basics and Introduction to Oracle9i PL/SQL* course.

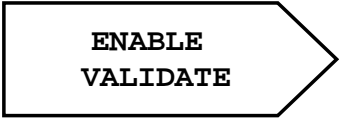
Enabling Constraints (continued)

Using Oracle Enterprise Manager to Define Constraints

From the OEM Console:

1. Navigate to Schema > [Schema name] > Tables.
2. Select the table in which the constraints are to be modified.
3. Open the Constraints tabbed page, and make the modifications.
4. Click Apply.

Enabling Constraints



ENABLE
VALIDATE

- Locks the table
- Can use unique or nonunique indexes
- Needs valid table data

```
ALTER TABLE hr.employees  
ENABLE VALIDATE CONSTRAINT emp_dept_fk;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Enabling Constraints

Enabling a `VALIDATE` constraint checks existing data for constraint violation. This is the default when a constraint is enabled. If executed when the constraint is disabled, it has the following effects:

- The table is locked and changes to the table are prevented until validation of existing data is complete.
- The Oracle server creates an index if one does not exist on the index columns. It creates a unique index while enabling a primary key or unique constraint that is nondeferrable. A nonunique index is built for a deferrable primary key or a unique constraint.

If this command is executed when a constraint is enforced, then it does not require any table locks during validation. The enforced constraint guarantees that no violations are introduced during validation. This has the following advantages:

- All constraints are enabled concurrently.
- Each constraint is internally parallelized.
- Concurrent activity on the table is permitted.

Enabling Constraints (continued)

Use the following command to enable an ENABLE VALIDATE constraint:

```
ALTER TABLE [ schema. ] table
    ENABLE [ VALIDATE ] {CONSTRAINT constraint
    | PRIMARY KEY| UNIQUE ( column [, column ] ... )}
    [ USING INDEX index_clause ]
    [ EXCEPTIONS INTO [ schema. ] table ]
```

Note

- The VALIDATE option is the default and does not need to be specified when enabling a constraint that is disabled.
- If data in the table violates the constraint, then the statement is rolled back and the constraint remains disabled.
- The use of the EXCEPTIONS clause is discussed in the following section.

Renaming Constraints

Use the following to rename a constraint:

```
ALTER TABLE employees  
  RENAME CONSTRAINT emp_dept_fk  
  TO employees_dept_fk;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Renaming Constraints

In Oracle9i Database Release 2, you now have the ability to rename constraints with the following syntax:

```
ALTER TABLE [schema.]table_name  
  RENAME CONSTRAINT old_constraint_name  
  TO new_constraint_name;
```

Renaming Constraints (continued)

Using Oracle Enterprise Manager to Rename a Constraint

From the OEM Console:

1. Navigate to Schema > [Schema Name] > Tables.
2. Select the table in which the constraint is to be modified.
3. Open the Constraints tabbed page, and type in the new constraint name.
4. Click Apply.

Using the EXCEPTIONS Table

- Create the EXCEPTIONS table by running the `utlexpt1.sql` script.
- Execute the ALTER TABLE statement with EXCEPTIONS option.
- Use subquery on EXCEPTIONS to locate rows with invalid data.
- Rectify the errors.
- Reexecute ALTER TABLE to enable the constraint.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Using the EXCEPTIONS Table

The EXCEPTIONS clause identifies any row that violates an enabled constraint. Use the following procedure to detect constraint violations, rectify them, and reenable a constraint:

1. If the EXCEPTIONS is not already created, run the `utlexpt1.sql` script:

```
SQL> @?/rdbms/admin/utlexpt1
Statement processed.
SQL> DESCRIBE exceptions
Name                Null?      Type
-----
ROW_ID              ROWID
OWNER                VARCHAR2(30)
TABLE_NAME           VARCHAR2(30)
CONSTRAINT           VARCHAR2(30)
```

Note: The exact name and location of the `utlexpt1.sql` script is specific to the operating system. For more information, see your operating system-specific Oracle documentation.

Using the EXCEPTIONS Table (continued)

2. Execute the ALTER TABLE command using the EXCEPTIONS clause:

```
SQL> ALTER TABLE hr.employee
      2 ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
      3 EXCEPTIONS INTO system.exceptions;
```

```
ALTER TABLE hr.employee
*
```

```
ORA-02298: cannot enable (HR.EMPLOYEE_DEPT_ID_FK) -
parent keys not found
```

If the EXCEPTIONS table is not qualified with the name of the owner, it must belong to the owner of the table that is being altered.

Rows are inserted into the EXCEPTIONS table. If you are rerunning the command, truncate the EXCEPTIONS table to remove all existing rows.

3. Identify invalid data by using a subquery on the EXCEPTIONS table:

```
SQL> SELECT rowid, id, last_name, dept_id
      2 FROM hr.employee
      3 WHERE ROWID in (SELECT row_id
      4 FROM exceptions)
      5 FOR UPDATE;

ROWID                                ID    LAST_NAME  DEPT_ID
-----                                -
AAAAeyAADAAAAA1AAA 1003 Pirie      50
1 row selected.
```

4. Correct the errors in the data:

```
SQL> UPDATE hr.employee
      2 SET dept_id=10
      3 WHERE rowid='AAAAeyAADAAAAA1AAA';
1 row processed.
SQL> COMMIT;
Statement processed.
```

Using the EXCEPTIONS Table (continued)

5. Truncate the EXCEPTIONS table and reenable the constraint:

```
SQL> TRUNCATE TABLE exceptions;
```

Statement processed.

```
SQL> ALTER TABLE hr.employee
```

```
2  ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
```

```
3  EXCEPTIONS INTO system.exceptions;
```

Statement processed.

Obtaining Constraint Information

Obtain information about constraints by querying the following views:

- DBA_CONSTRAINTS
- DBA_CONS_COLUMNS

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Obtaining Constraint Information

Use the following query to obtain the names, types, and statuses of all constraints on HR's EMPLOYEES table:

```
SQL> SELECT constraint_name, constraint_type, deferrable,
2 deferred, validated
3 FROM dba_constraints
4 WHERE owner='HR'
5 AND table_name='EMPLOYEES';
```

CONSTRAINT_NAME	C	DEFERRABLE	DEFERRED	VALIDATED
EMPLOYEE_DEPT..	R	DEFERRABLE	DEFERRED	VALIDATED
EMPLOYEE_ID_PK	P	DEFERRABLE	IMMEDIATE	VALIDATED
SYS_C00565	C	NOT DEFERRABLE	IMMEDIATE	VALIDATED

3 rows selected.

Obtaining Constraint Information (continued)

The following table shows the columns in the DBA_CONSTRAINTS view that are not self-evident:

Name	Description
CONSTRAINT_TYPE	The type of constraint is P if Primary Key, U if Unique, R if foreign key, or C if Check constraint. NOT NULL constraints are stored as check constraints.
SEARCH_CONDITION	Shows the condition specified for a check constraint
R_OWNER R_CONSTRAINT_NAME	Defines the owner and name of the referenced constraint for foreign keys
GENERATED	Indicates if the constraint name is system-generated (Valid values are USERNAME and GENERATED NAME.)
BAD	Indicates that the constraint is to be rewritten to avoid situations such as the Year 2000 problems
RELY	Is used in the optimizer, if this flag is set
LAST_CHANGE	Shows the date when the constraint was last enabled or disabled

Columns in constraints

To obtain the columns in the constraints on HR's EMPLOYEES table, use the following query:

```
SQL> SELECT c.constraint_name, c.constraint_type,  
2      cc.column_name  
3      FROM dba_constraints c, dba_cons_columns cc  
4      WHERE c.owner='HR'  
5      AND c.table_name='EMPLOYEE'  
6      AND c.owner = cc.owner  
7      AND c.constraint_name = cc.constraint_name  
8      ORDER BY cc.position;  
  
CONSTRAINT_NAME    C COLUMN_NAME  
-----  
EMPLOYEE_DEPT...  R DEPT_ID  
EMPLOYEE_ID_PK     P ID  
SYS_C00565         C LAST_NAME  
  
3 rows selected.
```

Obtaining Constraint Information (continued)

Finding primary key–foreign key relationships

To find foreign keys on HR's EMPLOYEE table and the parent constraints, use the following query:

```
SQL> SELECT c.constraint_name AS "Foreign Key",
  2  p.constraint_name AS "Referenced Key",
  3  p.constraint_type,
  4  p.owner,
  5  p.table_name
  6 FROM dba_constraints c, dba_constraints p
  7 WHERE c.owner='HR'
  8 AND c.table_name='EMPLOYEE'
  9 AND c.constraint_type='R'
 10 AND c.r_owner=p.owner
 11 AND c.r_constraint_name = p.constraint_name;
```

Foreign Key	Referenced Key	C OWNER	TABLE_NAME
-----	-----	-----	-----
EMPLOYEES_DEPT..	DEPT_PK	P HR	DEPARTMENT

1 row selected.

Summary

In this lesson, you should have learned how to:

- **Implement data integrity**
- **Use an appropriate strategy to create and maintain constraints**
- **Obtain constraint information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 13 Overview

This practice covers the following topics:

- **Creating constraints**
- **Enabling unique constraints**
- **Creating an `EXCEPTIONS` table**
- **Identifying existing constraint violations in a table, correcting the errors, and reenabling the constraints**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 13: Maintaining Data Integrity

- 1 Examine and run the script lab13_01.sql to create the constraints.
- 2 As user SYSTEM, query the data dictionary to:
 - a Check for constraints, whether they are deferrable, and their status.
Hint: Use the DBA_CONSTRAINTS view to get this information.
 - b Check the names and types of indexes created to validate the constraints.
Hint: The indexes are only created for primary key and unique constraints and have the same name as the constraints.
- 3 As user SYSTEM, run the script lab13_03.sql to insert two records into the PRODUCTS table.
- 4 Enable the unique constraint on the PRODUCTS table. Was it successful?
- 5
 - a Make sure that any new rows added to the table do not violate the constraint on the PRODUCTS table.
Hint: This can be done by enabling the NOVALIDATE constraint.
 - b Query the data dictionary to verify the effect of the change.
 - c Test that the constraint disables inserts that violate the change by adding a row with the following values:

PRODUCT_ID	PRODUCT_DESCRIPTION	LIST_PRICE
4000	Monitor	3000

- 6 Take the necessary steps to identify existing constraint violations in the PRODUCTS table, modify product codes as needed, and guarantee that all existing and new data does not violate the constraint. (Assume that the table has several thousand rows and it is too time-consuming to verify each row manually.)
Hint: Use the following steps:
 - a Create the EXCEPTIONS table.
 - b Run the command to enable the constraint and trap the exceptions.
 - c Use the ROWID in the EXCEPTIONS table to list the rows in the PRODUCTS table that violate the constraint. Do not list LOB columns.
 - d Rectify the errors.
 - e Enable the constraint.
- 7 Run the script lab13_07.sql to insert rows into the table. Were the inserts successful?
- 8 Now examine the script lab13_08. Notice that this script also performs the inserts in the same sequence. Run the script and check if it executes successfully.
- 9 Truncate the CUSTOMERS table. Was it successful?

14

Managing Password Security and Resources

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Objectives

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

- **Manage passwords using profiles**
- **Administer profiles**
- **Control use of resources using profiles**
- **Obtain password and resource limit information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Profiles

- A profile is a named set of password and resource limits.
- Profiles are assigned to users by the `CREATE USER` or `ALTER USER` command.
- Profiles can be enabled or disabled.
- Profiles can relate to the `DEFAULT` profile.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Profiles

A profile is a named set of the following password and resource limits:

- Password aging and expiration
- Password history
- Password complexity verification
- Account locking
- CPU time
- Input/output (I/O) operations
- Idle time
- Connect time
- Memory space (private SQL area for Shared Server only)
- Concurrent sessions

After a profile has been created, the database administrator can assign it to each user. If resource limits are enabled, the Oracle server limits the database usage and resources to the defined profile of the user.

Profiles (continued)

The Oracle server automatically creates a DEFAULT profile when the database is created.

The users who have not been explicitly assigned a specific profile conform to all the limits of the DEFAULT profile. All limits of the DEFAULT profile are initially unlimited.

However, the database administrator can change the values so that limits are applied to all users by default.

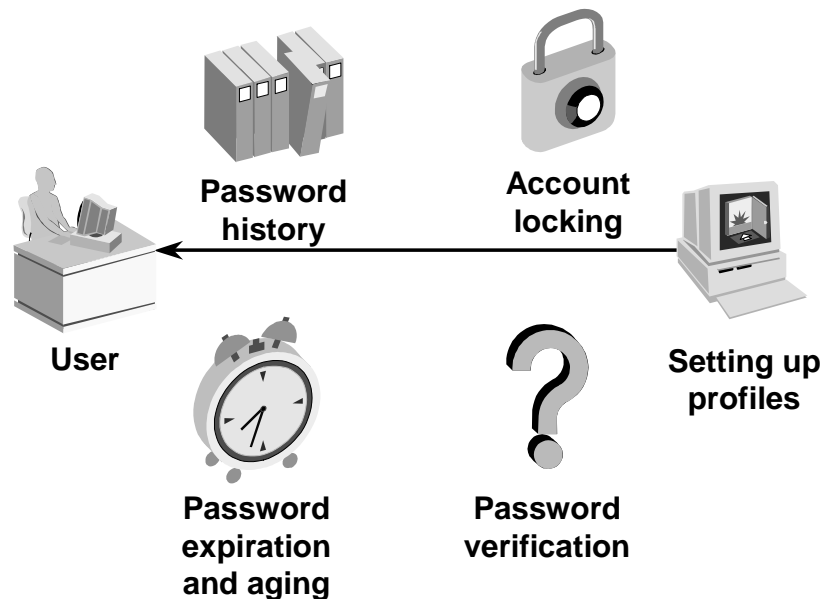
Profile usage

- Restrict users from performing some operations that require heavy use of resources.
- Make sure that users log off the database when they have left their session idle for some time.
- Enable group resource limits for similar users.
- Easily assign resource limits to users.
- Manage resource usage in large, complex multiuser database systems.
- Control the use of passwords

Profile characteristics

- Profile assignments do not affect current sessions.
- Profiles can be assigned only to users and not to roles or other profiles.
- If you do not assign a profile when creating a user, the user is automatically assigned the DEFAULT profile.

Password Management



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Password Management

For greater control over database security, Oracle password management is controlled by database administrators with profiles.

This lesson describes the available password management features:

- **Account locking:** Enables automatic locking of an account when a user fails to log in to the system in the specified number of attempts
- **Password aging and expiration:** Enables the password to have a lifetime, after which it expires and must be changed
- **Password history:** Checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes
- **Password complexity verification:** Performs a complexity check on the password to verify that it is complex enough to provide protection against intruders who might try to break into the system by guessing the password

Enabling Password Management

- Set up password management by using profiles and assigning them to users.
- Lock, unlock, and expire accounts using the `CREATE USER` or `ALTER USER` command.
- Password limits are always enforced.
- To enable password management, run the `utlpwdmg.sql` script as the user `SYS`.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Enabling Password Management

Create the profile to limit password settings, and assign the profile to the user by using the `CREATE USER` or `ALTER USER` command.

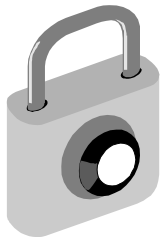
Password limit settings in profiles are always enforced.

When password management is enabled, the user account can be locked or unlocked by using the `CREATE USER` or `ALTER USER` command.

Note: Refer to the “Managing Users” lesson for details regarding the `CREATE USER` command.

Password Account Locking

Parameter	Description
<code>FAILED_LOGIN_ATTEMPTS</code>	Number of failed login attempts before lockout of the account
<code>PASSWORD_LOCK_TIME</code>	Number of days the account is locked after the specified number of failed login attempts



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Password Account Locking

The Oracle server automatically locks an account after the `FAILED_LOGIN_ATTEMPTS` value is reached. The account is either automatically unlocked after a specified time, defined by the `PASSWORD_LOCK_TIME` parameter, or it must be unlocked by the database administrator using the `ALTER USER` command.

The database account can be explicitly locked with the `ALTER USER` command. When this happens, the account is not automatically unlocked.

Note: The `ALTER USER` command will be demonstrated later in this lesson.

Password Expiration and Aging

Parameter	Parameter
PASSWORD_LIFE_TIME	Lifetime of the password in days after which the password expires
PASSWORD_GRACE_TIME	Grace period in days for changing the password after the first successful login after the password has expired



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Password Expiration and Aging

The `PASSWORD_LIFE_TIME` parameter sets the maximum lifetime after which the password must be changed.

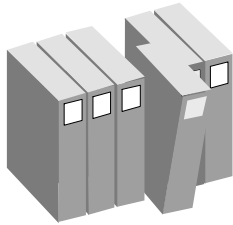
The database administrator can specify a `PASSWORD_GRACE_TIME` grace period that begins after the first attempt to log in to the database after password expiration. A warning message is generated every time the user tries to log in until the grace period is over. The user is expected to change the password within the grace period.

If the password is not changed, the account is locked.

The user's account status is changed to `EXPIRED` by explicitly setting the password to be expired.

Password History

Parameter	Description
PASSWORD_REUSE_TIME	Number of days before a password can be reused
PASSWORD_REUSE_MAX	Maximum number of changes required before a password can be reused



ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Password History

Password history checks ensure that a user cannot reuse a password for a specified time interval. These checks can be implemented by using one of the following:

- **PASSWORD_REUSE_TIME**: Specifies that a user cannot reuse a password for a given number of days
- **PASSWORD_REUSE_MAX**: Forces a user to define a password that is not identical to earlier passwords

When one parameter is set to a value other than **DEFAULT** or **UNLIMITED**, the other parameter must be set to **UNLIMITED**.

Password Verification

Parameter	Description
PASSWORD_VERIFY_FUNCTION	PL/SQL function that performs a password complexity check before a password is assigned

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Password Verification

Before assigning a new password to a user, a PL/SQL function can be invoked to verify the validity of the password.

The Oracle server provides a default verification routine or the database administrator can write a PL/SQL function.

User-Provided Password Function

This function must be created in the `SYS` schema and must have the following specification:

```
function_name(  
  userid_parameter IN VARCHAR2(30),  
  password_parameter IN VARCHAR2(30),  
  old_password_parameter IN VARCHAR2(30))  
RETURN BOOLEAN
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

User-Provided Password Function

When a new password verification function is added, the database administrator must consider the following restrictions:

- The procedure must use the specification indicated in the slide.
- The procedure returns the value `True` for success and `False` for failure.
- If the password function raises an exception, then an error is returned and the `ALTER USER` or `CREATE USER` command is terminated.
- The password function is owned by `SYS`.
- If the password function becomes invalid, then an error message is returned and the `ALTER USER` or `CREATE USER` command is terminated.

Note: Refer to the “Managing Users” lesson for details regarding `CREATE USER`.

Password Verification Function

VERIFY_FUNCTION

- Minimum length is four characters.
- Password should not be equal to username.
- Password should have at least one alphabetic, one numeric, and one special character.
- Password should differ from the previous password by at least three letters.

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Password Verification Function

The Oracle server provides a complexity verification function, in the form of a default PL/SQL function called `VERIFY_FUNCTION` of the `utlpwdmg.sql` script, which must be run in the `SYS` schema.

During the execution of the `utlpwdmg.sql` script, the Oracle server creates `VERIFY_FUNCTION` and changes the `DEFAULT` profile with the following `ALTER PROFILE` command:

```
SQL> ALTER PROFILE DEFAULT LIMIT
2  PASSWORD_LIFE_TIME 60
3  PASSWORD_GRACE_TIME 10
4  PASSWORD_REUSE_TIME 1800
5  PASSWORD_REUSE_MAX UNLIMITED
6  FAILED_LOGIN_ATTEMPTS 3
7  PASSWORD_LOCK_TIME 1/1440
8  PASSWORD_VERIFY_FUNCTION verify_function;
```

Creating a Profile: Password Settings

```
CREATE PROFILE grace_5 LIMIT
  FAILED_LOGIN_ATTEMPTS 3
  PASSWORD_LOCK_TIME UNLIMITED
  PASSWORD_LIFE_TIME 30
  PASSWORD_REUSE_TIME 30
  PASSWORD_VERIFY_FUNCTION verify_function
  PASSWORD_GRACE_TIME 5;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Profile

Use the following CREATE PROFILE command to administer passwords:

```
CREATE PROFILE profile LIMIT
  [FAILED_LOGIN_ATTEMPTS max_value]
  [PASSWORD_LIFE_TIME max_value]
  [ {PASSWORD_REUSE_TIME
    |PASSWORD_REUSE_MAX} max_value]
  [PASSWORD_LOCK_TIME max_value]
  [PASSWORD_GRACE_TIME max_value]
  [PASSWORD_VERIFY_FUNCTION
    {function|NULL|DEFAULT} ]
```

Creating a Profile (continued)

where

PROFILE: Is the name of the profile to be created

FAILED_LOGIN_ATTEMPTS: Specifies the number of failed attempts to log in to the user account before the account is locked

PASSWORD_LIFE_TIME: Limits the number of days the same password can be used for authentication. The password expires if not changed within this period, and further connections are rejected.

PASSWORD_REUSE_TIME: Specifies the number of days before a password can be reused. If you set **PASSWORD_REUSE_TIME** to an integer value, then you must set **PASSWORD_REUSE_MAX** to **UNLIMITED**.

PASSWORD_REUSE_MAX: Specifies the number of password changes required before the current password can be reused. Prior to Oracle9i, if you set **PASSWORD_REUSE_MAX** to an integer value, then you must set **PASSWORD_REUSE_TIME** to **UNLIMITED**.

PASSWORD_LOCK_TIME: Specifies the number of days an account will be locked after the specified number of consecutive failed login attempts

PASSWORD_GRACE_TIME: Specifies the number of days after the grace period begins during which a warning is issued and login is allowed. The password expires if not changed during grace period.

PASSWORD_VERIFY_FUNCTION: Enables a PL/SQL password complexity verification function to be passed as an argument to the **CREATE PROFILE** statement

Creating a Profile (continued)

Using Oracle Enterprise Manager to Create a Profile

From the OEM Console:

1. Navigate to Security > Profiles.
2. Select Create from the right-mouse menu.
3. Enter a name for the Profile and complete other fields or accept the default values.
4. Open the Password tabbed page and enter the account password parameters.
5. Click Create.

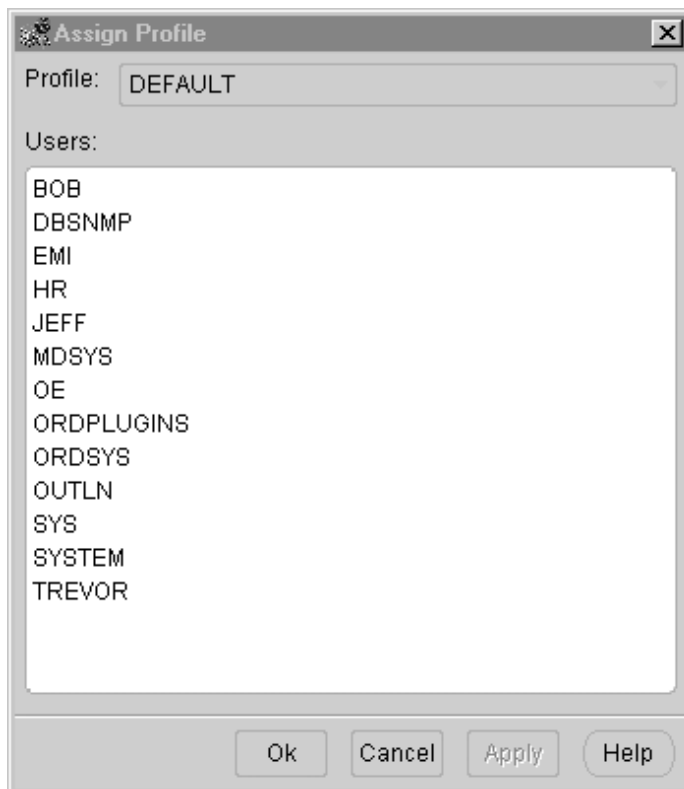
The screenshot shows the 'Create Profile' dialog box with the title 'Create Profile - sys@DBA01_STC-SUN01.US.ORACLE.COM'. It features two tabs: 'General' and 'Password'. The 'General' tab is selected, displaying a 'Profile Name' text field and a 'Details' section with four rows of settings: CPU/Session, CPU/Call, Connect Time, and Idle Time. Each row has a dropdown menu set to 'Default' and a unit label (Sec./100, Sec./100, Minutes, Minutes). Below this is a 'Database Services' section with five rows: Concurrent Sessions, Reads/Session, Reads/Call, Private SGA, and Composite Limit. Each row has a dropdown menu set to 'Default' and a unit label (Per User, Blocks, Blocks, KBytes, Service Units). At the bottom of the dialog are four buttons: 'Create', 'Cancel', 'Show SQL', and 'Help'.

Creating a Profile (continued)

Using Oracle Enterprise Manager to Create a Profile

From the OEM Console:

1. Navigate to Security > Profiles.
2. Select Assign a Profile to User(s) from the right-mouse menu.
3. Select the user(s).
4. Click OK.



Altering a Profile: Password Setting

Use **ALTER PROFILE** to change password limits.

```
ALTER PROFILE default LIMIT  
FAILED_LOGIN_ATTEMPTS 3  
PASSWORD_LIFE_TIME 60  
PASSWORD_GRACE_TIME 10;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Altering a Profile

Use the **ALTER PROFILE** command to change the password limits assigned to a profile:

```
ALTER PROFILE profile LIMIT  
[FAILED_LOGIN_ATTEMPTS max_value]  
[PASSWORD_LIFE_TIME max_value]  
[ {PASSWORD_REUSE_TIME  
  |PASSWORD_REUSE_MAX} max_value]  
[PASSWORD_LOCK_TIME max_value]  
[PASSWORD_GRACE_TIME max_value]  
[PASSWORD_VERIFY_FUNCTION  
  {function|NULL|DEFAULT} ]
```

To set the password parameters to less than a day:

1 hour: `PASSWORD_LOCK_TIME = 1/24`

10 minutes: `PASSWORD_LOCK_TIME = 10/1440`

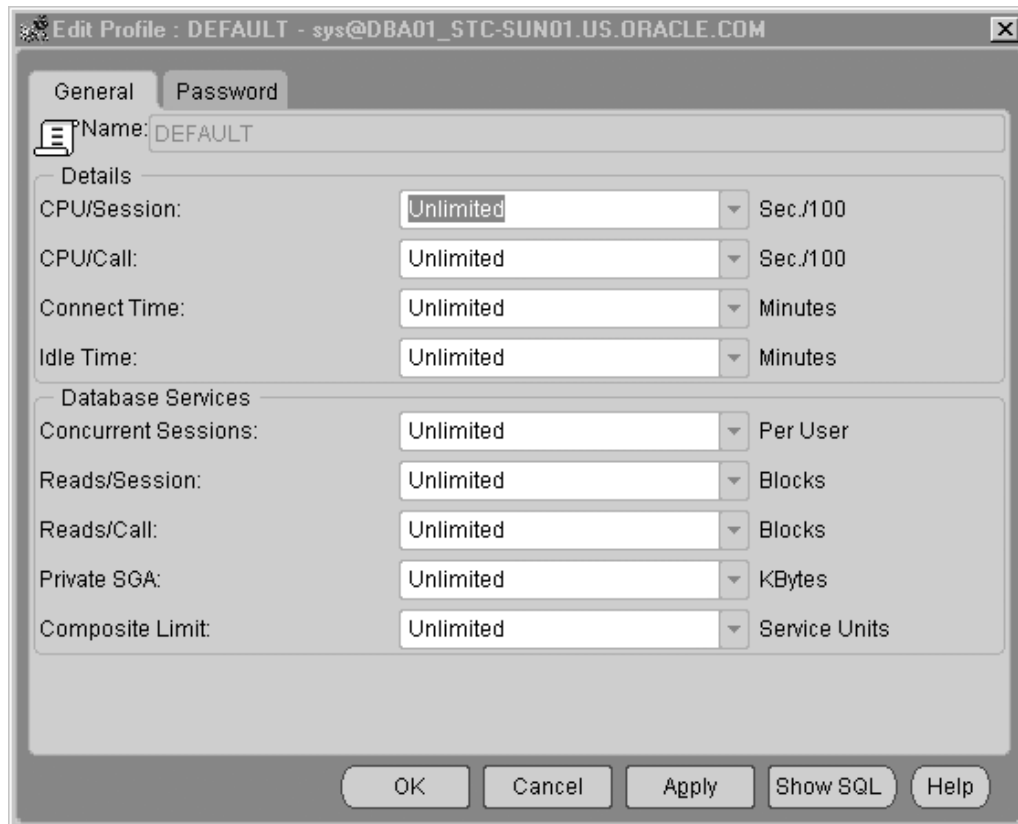
5 minutes: `PASSWORD_LOCK_TIME = 5/1440`

Altering a Profile (continued)

Using Oracle Enterprise Manager to Alter a Profile

From the OEM Console:

1. Navigate to Security > Profiles.
2. Highlight the profile.
3. Open the Password tabbed page and alter the profile.
4. Click Apply.



Dropping a Profile: Password Setting

- Drop the profile using `DROP PROFILE` command.
- `DEFAULT` profile cannot be dropped.
- `CASCADE` revokes the profile from the user to whom it was assigned.

```
DROP PROFILE developer_prof;
```

```
DROP PROFILE developer_prof CASCADE;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Dropping a Profile: Password Setting

Drop a profile using the `DROP PROFILE` command:

```
DROP PROFILE profile [CASCADE]
```

where:

`profile`: Is the name of the profile to be dropped

`CASCADE`: Revokes the profile from users to whom it is assigned. (The Oracle server automatically assigns the `DEFAULT` profile to such users. Specify this option to drop a profile that is currently assigned to users.)

Guidelines

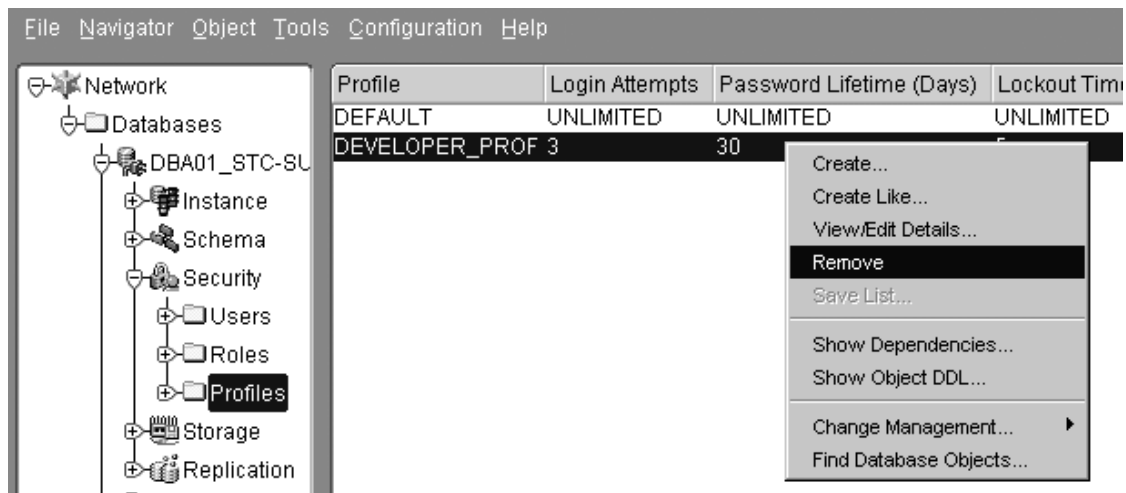
- The `DEFAULT` profile cannot be dropped.
- When a profile is dropped, this change applies only to sessions that are created subsequently and not to the current sessions.

Dropping a Profile: Password Setting (continued)

Using Oracle Enterprise Manager to Drop a Profile

From the OEM Console:

1. Navigate to Security > Profiles.
2. Highlight the profile.
3. Select Remove from the right-mouse menu.
4. Click Yes to confirm drop.



Resource Management

- Resource management limits can be enforced at the session level, the call level, or both.
- Limits can be defined by profiles using the `CREATE PROFILE` command.
- Enable resource limits with the:
 - `RESOURCE_LIMIT` initialization parameter
 - `ALTER SYSTEM` command

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Resource Management

Use the following steps to control the usage of resources with profiles:

1. Create a profile with the `CREATE PROFILE` command to determine the resource and password limits.
2. Assign profiles with the `CREATE USER` or `ALTER USER` command.
3. Enforce resource limits with the `ALTER SYSTEM` command or by editing the initialization parameter file (and stopping and restarting the instance).

These steps are discussed in detail in the following section.

Note: Enforcing the resource limits is not required for enabling Oracle password management.

Enabling Resource Limits

- Set the initialization parameter `RESOURCE_LIMIT` to `TRUE`.
- Enforce the resource limits by enabling the parameter with the `ALTER SYSTEM` command.

```
ALTER SYSTEM SET RESOURCE_LIMIT=TRUE;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Enabling Resource Limits

Enable or disable the enforcement of resource limits by altering the `RESOURCE_LIMIT` initialization parameter or by using the `ALTER SYSTEM` command.

RESOURCE_LIMIT initialization parameter

- To enable or disable enforcement of resource limits, alter this parameter in the initialization file and restart the instance.
- A value of `True` enables enforcement.
- A value of `False` disables enforcement (default).
- Use this parameter to enable enforcement architecture.

ALTER SYSTEM command

- To enable or disable enforcement of resource limits for an instance, use the `ALTER SYSTEM` command.
- The setting that is specified using the `ALTER SYSTEM` command remains in effect until altered again or until the database is shut down.
- Use this command to enable or to disable enforcement when the database cannot be shut down.

Setting Resource Limits at Session Level

Resource	Description
CPU_PER_SESSION	Total CPU time measured in hundredths of seconds
SESSIONS_PER_USER	Number of concurrent sessions allowed for each username
CONNECT_TIME	Elapsed connect time measured in minutes
IDLE_TIME	Periods of inactive time measured in minutes
LOGICAL_READS_PER_SESSION	Number of data blocks (physical and logical reads)
PRIVATE_SGA	Private space in the SGA measured in bytes (for Shared Server only)

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Setting Resource Limits at Session Level

Guidelines

Profile limits can be enforced at the session level, the call level, or both. Session-level limits are enforced for each connection.

When a session-level limit is exceeded:

- An error message returns. For example:
ORA-02391: exceeded simultaneous SESSION_PER_USER limit
- The Oracle server disconnects the user.

Guidelines

- IDLE_TIME: Is calculated for the server process only. It does not take into account application activity. The IDLE_TIME limit is not affected by long-running queries and other operations.
- LOGICAL_READS_PER_SESSION: Is a limitation on the total number of reads from both memory and disk. This can be done to ensure that no I/O intensive statements hoard memory and tie up the disk.
- PRIVATE_SGA: Applies only when running the Shared Server architecture and can be specified in MB or KB.

Note: The shared server architecture is covered in detail in the *Oracle9i Database Administration Fundamentals II* course.

Setting Resource Limits at Call Level

Resource	Description
CPU_PER_CALL	CPU time per call in hundredths of seconds
LOGICAL_READS_PER_CALL	Number of data blocks that can be read per call

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Setting Resource Limits at Call Level

Call-level limits are enforced for each call that is made while executing a SQL statement.

When a call-level limit is exceeded:

- The processing of the statement is halted.
- The statement is rolled back.
- All previous statements remain intact.
- The user's session remains connected.

Creating a Profile: Resource Limit

```
CREATE PROFILE developer_prof LIMIT
SESSIONS_PER_USER 2
CPU_PER_SESSION 10000
IDLE_TIME 60
CONNECT_TIME 480;
```

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Creating a Profile: Resource Limit

Create a profile using the following CREATE PROFILE command:

```
CREATE PROFILE profile LIMIT
[SESSIONS_PER_USER max_value]
[CPU_PER_SESSION max_value]
[CPU_PER_CALL max_value]
[CONNECT_TIME max_value]
[IDLE_TIME max_value]
[LOGICAL_READS_PER_SESSION max_value]
[LOGICAL_READS_PER_CALL max_value]
[COMPOSITE_LIMIT max_value]
[PRIVATE_SGA max_bytes]
```

where:

profile: The name of the profile

max_value: An integer, UNLIMITED, or DEFAULT

max_bytes: An integer optionally followed by KB or MB UNLIMITED, or DEFAULT

Creating a Profile: Resource Limit (continued)

UNLIMITED: Indicates that a user assigned this profile can use an unlimited amount of this resource

DEFAULT: Indicates this profile is subject to the limit for this resource, as specified in the DEFAULT profile

COMPOSITE_LIMIT: Limits the total resource cost for a session expressed in service units. Oracle calculates the resource cost as a sum of:

- CPU_PER_SESSION
- CONNECT_TIME
- LOGICAL_READS_PER_SESSION
- PRIVATE_SGA

The RESOURCE_COST data dictionary view provides resource limits assigned to different resources.

Note: Refer to the *Oracle9i SQL Reference* document for information about how to specify the weight for each session resource ALTER RESOURCE COST command.

Creating a Profile: Resource Limit (continued)

Using Oracle Enterprise Management to Set Resource Limits

From the OEM Console:

- Navigate to Security > Profiles.
- Select Create from the right-mouse menu.
- Enter the resource parameters in the General tabbed page.
- Click Create.

Create Profile - sys@DBA01_STC-SUN01.US.ORACLE.COM

General Password

Profile Name: DEVELOPER_PROF

Details

CPU/Session:	1000	Sec./100
CPU/Call:	1000	Sec./100
Connect Time:	120	Minutes
Idle Time:	60	Minutes

Database Services

Concurrent Sessions:	2	Per User
Reads/Session:	5000	Blocks
Reads/Call:	5000	Blocks
Private SGA:	16	KBytes
Composite Limit:	5000000	Service Units

Create Cancel Show SQL Help

Managing Resources Using the Database Resource Manager

- Provides the Oracle server with more control over resource management decisions
- Elements of the Database Resource Manager:
 - Resource consumer group
 - Resource plan
 - Resource allocation method
 - Resource plan directives
- Uses the `DBMS_RESOURCE_MANAGER` package to create and maintain elements
- Requires `ADMINISTER_RESOURCE_MANAGER` privilege

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Resources Using Database Resource Manager

The goal of the Database Resource Manager is to give the Oracle server more control over resource management decisions, thus circumventing problems that result from inefficient operating system management.

Elements of the Database Resource Manager

Resource consumer group: Groups of users, or sessions, grouped together based on resource processing requirements

Resource plan: Contains directives that specify how resources are allocated to resource consumer groups

Resource allocation method: Used by the Database Resource Manager when allocating for a particular resource

Resource plan directive: Used by administrators to associate resource consumer groups with particular plans and allocate resources among resource consumer groups

Administering the Database Resource Manager

You must have the `ADMINISTER_RESOURCE_MANAGER` system privilege to administer the Database Resource Manager (`DBMS_RESOURCE_MANAGER`). Typically, DBAs have this privilege with the `ADMIN` option as part of the DBA role.

Managing Resources Using the Database Resource Manager

- **Resource plans specify the resource consumer groups belonging to the plan.**
- **Resource plans contain directives for how to allocate resources among consumer groups.**

ORACLE

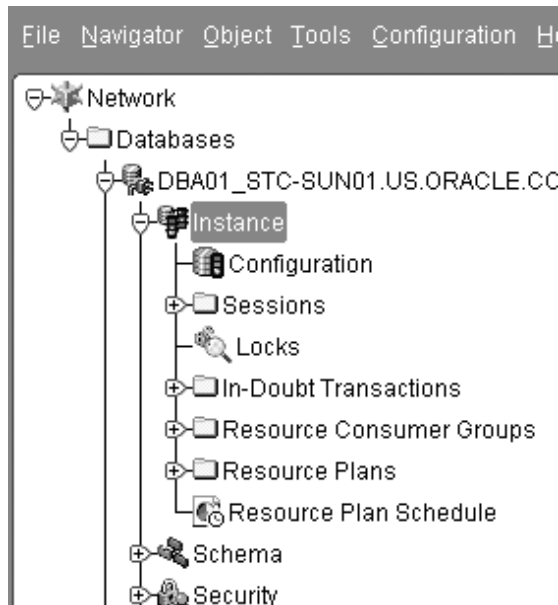
Copyright © Oracle Corporation, 2002. All rights reserved.

Managing Resources Using the Database Resource Manager

Using Oracle Enterprise Manager to Set Resource Manager

From the OEM Console:

- Navigate to Instance
 - Select Resource Consumer Groups to create or modify.
 - Select Resource Plans to create or modify.
 - Select Resource Plan Schedule to create or modify.



Resource Plan Directives

The Database Resource Manager provides several means of allocating resources:

- CPU method
- Active session pool and queuing
- Degree of parallelism limit
- Automatic consumer group switching
- Maximum estimated execution time
- Undo quota

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Resource Plan Directives

CPU method: Allows you to specify how CPU resources are to be allocated among consumer groups

Active session pool with queuing: You can control the maximum number of concurrently active sessions allowed within a consumer group. This maximum designates the active session pool. When the session cannot be initiated because the pool is full, the session is placed into a queue. When an active session completes, the first session in the queue is scheduled for execution. A timeout period can also be defined so that a job in the queue will timeout, causing it to abort with an error.

Degree of parallelism limit: Specifies a parallel degree limit for any operation within a consumer group

Automatic consumer group switching: Allows you to control resources by specifying certain criteria. If the criteria are not met, this causes the automatic switching of sessions to another consumer group. The following criteria are used to determine switching:

Switch group: Group being switched to

Switch time: Switch time in seconds

Switch estimate: An estimate of how long the operation will take to complete that is used to decide whether to switch an operation even before it starts

Resource Plan Directives (continued)

Maximum estimated execution time: Estimates the execution time of an operation proactively. A DBA can define the maximum estimated execution time required for any operation at any given time by setting the resource plan directive parameter. `MAX_ESTIMATED_EXEC_TIME`. If the operation's estimate is more than the `MAX_ESTIMATED_EXEC_TIME` defined, the operation will not start, thereby eliminating the exceptionally large job that would use excessive system resources.

Undo Pool: An undo pool for each consumer group can be specified to control the amount of total undo that can be generated by a consumer group. When a consumer group exceeds its limit, the current DML statement that is generating the redo is terminated. The undo pool is defined by the resource plan directive parameter called `UNDO_POOL`.

Note: The Database Resource Manager is covered further in the *Oracle9i Performance Tuning* course.

Obtaining Password and Resource Limit Information

Information about password and resource limits can be obtained by querying the following views:

- DBA_USERS
- DBA_PROFILES

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Obtaining Password and Resource Limit Information

Use the DBA_USERS view to obtain information about account status.

```
SQL> SELECT username, password, account_status,  
2    FROM dba_users;
```

USERNAME	PASSWORD	ACCOUNT_STATUS
-----	-----	-----
SYS	8A8F025737A9097A	OPEN
SYSTEM	D4DF7931AB130E37	OPEN
OUTLN	4A3BA55E08595C81	OPEN
DBSNMP	E066D214D5421CCC	OPEN
HR	BB69FBB77CFA6B9A	OPEN
OE	957C7EF29CC223FC	LOCKED

Obtaining Password and Resource Limit Information (continued)

Query the DBA_PROFILES view to display password profile information:

```
SQL> SELECT * FROM dba_profiles
      2  WHERE resource_type='PASSWORD'
      3  AND profile='GRACE_5';
```

PROFILE RESOURCE_NAM	RESOURCE	LIMIT
-----	-----	-----
GRACE_5 FAILED_LOGIN_ATTEMPTS	PASSWORD	3
GRACE_5 PASSWORD_LIFE_TIME	PASSWORD	30
GRACE_5 PASSWORD_REUSE_TIME	PASSWORD	30
GRACE_5 PASSWORD_REUSE_MAX	PASSWORD	UNLIMITED
GRACE_5 PASSWORD_VERIFY_FUNCTION	PASSWORD	DEFAULT
GRACE_5 PASSWORD_LOCK_TIME	PASSWORD	UNLIMITED
GRACE_5 PASSWORD_GRACE_TIME	PASSWORD	5

Summary

In this lesson, you should have learned how to:

- **Administer passwords**
- **Administer profiles**
- **Obtain password and resource limit information**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 14 Overview

This practice covers the following topics:

- **Enabling password management**
- **Defining profiles and assigning to users**
- **Disabling password management**

ORACLE

Copyright © Oracle Corporation, 2002. All rights reserved.

Practice 14 Overview

Note: Practice can be accomplished using SQL*Plus or using Oracle Enterprise Manager and SQL*Plus Worksheet.

Practice 14: Managing Password Security and Resources

- 1 **a** Run the `lab14_01.sql` script to create user `Jeff`.
b Connect as user `SYS` and enable password management by running the `$ORACLE_HOME/rdbms/admin/utlpwdmg.sql` script.
- 2 Try to change the password for user `Jeff` to `Jeff`. What happens?
- 3 Try changing the password for `Jeff` to follow the password management format.
Hint: Password should contain at least one digit, one character, and one punctuation.
- 4 Alter the `DEFAULT` profile to ensure the following applies to users assigned the `DEFAULT` profile:
 - After two login attempts, the account should be locked.
 - The password should expire after 30 days.
 - The same password should not be reused for at least one minute.
 - The account should have a grace period of five days to change an expired password.
 - Make sure that these requirements have been implemented.**Hints:**
 - Use the `ALTER PROFILE` command to change the default profile limits.
 - Query the `DBA_PROFILES` data dictionary view to verify the result.
- 5 Log in as user `Jeff` supplying an invalid password. Try this twice, then log in again, this time supplying the correct password. What happens?
- 6 Using the `DBA_USERS` data dictionary view, verify that user `Jeff` is locked. Unlock the account for user `Jeff`. After unlocking user `Jeff`, connect as `Jeff`.
Hint: Execute the `ALTER USER` command to unlock the account.
- 7 Connect as user `SYS` and disable password checks for the `DEFAULT` profile.
Hint: Execute the `ALTER PROFILE` command to disable the password checks.
- 8 Log in to user `Jeff` supplying an invalid password. Try this twice, then log in again, this time supplying the correct password. What happens? Why?

