Oracle7™ Server Administrator’s Guide

Release 7.3

February, 1996

Part No. A32535–1

ORACLE®

Oracle7™ Server Administrator's Guide, Release 7.3
Part No. A32535–1

Author: Joyce Fee

Graphic Designer: Valarie Moore

Contributors: John Bellemore, Andrea Borr, Bill Bridge, Gray Clossman, Jeff Cohen, Ahmed Ezzat, John Frazzini, Gary Hallmark, Bhaskar Himatsingka, Alex Ho, Ken Jacobs, Sandeep Jain, Robert Jenkins Jr., Valerie Kane, Jonathan Klein, Phil Locke, Brom Mahbod,William Maimone, Andrew Mendelsohn, Gary Ngai, Greg Pongracz, Maria Pratt, Mary Rhodes, Hari Sankar, Marc Simon, Lynne Thieme, Alex Tsukerman.

This software was not developed for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It is the customer's responsibility to take all appropriate measures to ensure the safe use of such applications if the programs are used for such purposes.

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error–free.

Preface

This Guide describes how to manage the Oracle7 Server, a relational database management system (RDBMS).

Information in this Guide applies to the Oracle7 Server, running on all operating systems. It provides information about the base Oracle7 Server. This Guide also refers to other manuals that describe special options, including the following:

- Distributed Option
- Parallel Server Option
- Advanced Replication Option

Audience

This Guide is for people who administer the operation of an Oracle database system. These people, referred to as "database administrators" (DBAs), are assumed to be responsible for ensuring the smooth operation of an Oracle database system and for monitoring its use. The responsibilities of database administrators are described in Chapter 1.

Knowledge Assumed of the Reader

Readers of this Guide are assumed to be familiar with relational database concepts. They are also assumed to be familiar with the operating system environment under which they are running Oracle.

As a prerequisite, **all readers should read the first chapter of** *Oracle7 Server Concepts*, "A Technical Introduction to the Oracle7 Server". This chapter is a comprehensive introduction to the concepts and terminology used throughout this Guide.

Readers Interested in Installation and Migration Information

Administrators frequently participate in installing the Oracle7 Server software and migrating existing Oracle databases to newer formats (for example, Version 6 databases to Oracle7 format). This Guide is not an installation or migration manual.

If your primary interest is installation, see your operating system–specific Oracle documentation.

If your primary interest is database or application migration, see the *Oracle7 Server Migration* manual.

Readers Interested in Application Design Information

In addition to administrators, experienced users of Oracle and advanced database application designers might also find information in this Guide useful.

However, database application developers should also see the *Oracle7 Server Application Developer's Guide* and the documentation for the tool or language product they are using to develop Oracle database applications.

How to Use This Guide

Every reader of this Guide **must** read Chapter 1 of the *Oracle7 Server Concepts* manual, “Introduction to the Oracle7 Server.” This overview of the concepts and terminology related to Oracle7 provides a foundation for the more detailed information in this Guide. The rest of the *Oracle7 Server Concepts* manual explains the Oracle7 architecture and features, and how they operate in more detail.

Conventions Used in This Guide

The following sections explain the conventions used in this Guide.

Text of the Guide

The following section explains the conventions used within the text of this Guide:

UPPERCASE WORDS

Uppercase text is used to call attention to command keywords, object names, parameters, filenames, and so on. For example:

If you create a private rollback segment, the name of the rollback segment must be included in the ROLLBACK_SEGMENTS parameter of the parameter file.

Italicized Words

Italicized words within text are used to indicate the first occurrence and definition of a term, as in the following example:

A *database* is a collection of data to be treated as a unit. The general purpose of a database is to store and retrieve related information, as needed.

Italicized words are also used to indicate emphasis, book titles, and to highlight names of performance statistics.

Examples of the Server Manager Interface

This Guide provides examples of the dialog boxes and menus of Server Manager, your primary utility for managing an Oracle database. Illustrations show the character mode Server Manager screen. However, the actual appearance of your screen may differ, depending on your system’s user interface.

For more information on Server Manager, see the *Oracle Server Manager User’s Guide*.

Examples of Commands and Statements

SQL, Server Manager, and SQL*Plus commands and statements appear separated from the text of paragraphs in a fixed–width font:

```
ALTER TABLESPACE users   ADD DATAFILE ’users2.ora’ SIZE 50K;
```

Punctuation , ’ ”

Example statements may include punctuation such as commas or quotation marks. All punctuation given in example statements is required. All example statements are terminated with a semicolon. Depending on the application being used, a semicolon or other terminator may or may not be required to end a statement.

Uppercase Words
INSERT, SIZE

Uppercase words in example statements are used to indicate the keywords within Oracle SQL. However, note that when issuing statements, keywords are not case–sensitive.

Lowercase Words
emp, users2.ora

Lowercase words in example statements are used to indicate words supplied only for the context of the example. For example, lowercase words may indicate the name of a table, column, or file. Some operating systems are case sensitive, so refer to your installation or user’s guide to determine whether you must pay attention to case.

Your Comments Are Welcome

We value and appreciate your comments as an Oracle user and reader of the manuals. As we write, revise, and evaluate our documentation, your opinions are the most important input we receive. At the back of our printed manuals is a Reader’s Comment Form, which we encourage you to use to tell us what you like and dislike about this manual or other Oracle manuals. If the form is not available, please use the following address or FAX number.

Oracle7 Server Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood City, CA 94065
U.S.A.
FAX: 415–506–7200

Contents

PART I **BASIC DATABASE ADMINISTRATION**

Chapter 1 The Oracle7 Database Administrator . . . . 1 – 1
Types of Oracle7 Users . . . . 1 – 2
Database Administrators . . . . 1 – 2
Security Officers . . . . 1 – 3
Application Developers . . . . 1 – 3
Application Administrators . . . . 1 – 3
Database Users . . . . 1 – 3
Network Administrators . . . . 1 – 3
Database Administrator Security and Privileges . . . . 1 – 4
The Database Administrator’s Operating System Account . . . . 1 – 4
Database Administrator Usernames . . . . 1 – 4
The DBA Role . . . . 1 – 5
Database Administrator Authentication . . . . 1 – 6
Selecting an Authentication Method . . . . 1 – 6
Using Operating System Authentication . . . . 1 – 7
OSOPER and OSDBA . . . . 1 – 7
Using an Authentication Password File . . . . 1 – 8
Password File Administration . . . . 1 – 9
Using ORAPWD . . . . 1 – 9
Setting REMOTE_LOGIN_ PASSWORDFILE . . . . 1 – 11
Adding Users to a Password File . . . . 1 – 12
Connecting with Administrator Privileges . . . . 1 – 14
Maintaining a Password File . . . . 1 – 15

Database Administrator Utilities . . . . 1 – 16
Server Manager . . . . 1 – 16
SQL*Loader . . . . 1 – 17
Export and Import . . . . 1 – 17
Initial Priorities of a Database Administrator . . . . 1 – 17
Step 1: Install the Oracle7 Software . . . . 1 – 18
Step 2: Evaluate the Database Server Hardware . . . . 1 – 18
Step 3: Plan the Database . . . . 1 – 18
Step 4: Create and Open the Database . . . . 1 – 19
Step 5: Implement the Database Design . . . . 1 – 19
Step 6: Back up the Database . . . . 1 – 19
Step 7: Enroll System Users . . . . 1 – 20
Step 8: Tune Database Performance . . . . 1 – 20
Identifying Oracle Software Releases . . . . 1 – 20
Release Number Format . . . . 1 – 20
Versions of Other Oracle Software . . . . 1 – 21
Checking Your Current Release Number . . . . 1 – 21

Chapter 2 **Creating a Database . . . . 2 – 1**
Considerations Before Creating a Database . . . . 2 – 2
Creation Prerequisites . . . . 2 – 3
Using an Initial Database . . . . 2 – 3
Migrating an Older Version of the Database . . . . 2 – 3
Creating an Oracle7 Database . . . . 2 – 4
Steps for Creating an Oracle7 Database . . . . 2 – 4
Creating a Database: Example . . . . 2 – 7
Troubleshooting Database Creation . . . . 2 – 8
Dropping a Database . . . . 2 – 8
Parameters . . . . 2 – 9
DB_NAME and DB_DOMAIN . . . . 2 – 9
CONTROL_FILES . . . . 2 – 10
DB_BLOCK_SIZE . . . . 2 – 11
DB_BLOCK_BUFFERS . . . . 2 – 11
PROCESSES . . . . 2 – 12
ROLLBACK_SEGMENTS . . . . 2 – 12
License Parameters . . . . 2 – 12
LICENSE_MAX_
SESSIONS and LICENSE_SESSIONS_WARNING . . . . 2 – 12
LICENSE_MAX_USERS . . . . 2 – 13
Considerations After Creating a Database . . . . 2 – 14
Initial Tuning Guidelines . . . . 2 – 14
Allocating Rollback Segments . . . . 2 – 15

Choosing the Number of DB_BLOCK_LRU_ LATCHES . . . 2 – 15
Distributing I/O . . . . 2 – 16

Chapter 3 **Starting Up and Shutting Down . . . . 3 – 1**
Startup Procedures . . . . 3 – 2
Preparing to Start an Instance . . . . 3 – 2
Starting an Instance: Scenarios . . . . 3 – 3
Altering Database Availability . . . . 3 – 6
Mounting a Database to an Instance . . . . 3 – 6
Opening a Closed Database . . . . 3 – 7
Restricting Access to an Open Database . . . . 3 – 7
Shutdown Procedures . . . . 3 – 8
Shutting Down a Database Under Normal Conditions . . . . . 3 – 8
Shutting Down a Database Immediately . . . . 3 – 9
Aborting an Instance . . . . 3 – 9
Using Parameter Files . . . . 3 – 10
The Sample Parameter File . . . . 3 – 10
The Number of Parameter Files . . . . 3 – 10
The Location of the Parameter File in Distributed Environments . . . . 3 – 11

PART II **ORACLE SERVER CONFIGURATION**

Chapter 4 **Managing Oracle7 Processes . . . . 4 – 1**
Configuring Oracle7 for Dedicated Server Processes . . . . 4 – 2
When to Connect to a Dedicated Server Process . . . . 4 – 3
Configuring Oracle7 for Multi–Threaded Server Processes . . . . 4 – 3
SHARED_POOL_ SIZE: Allocating Additional Space in the Shared Pool for Shared Server . . . . 4 – 5
MTS_LISTENER_ ADDRESS: Setting the Listener Process Address . . . . 4 – 5
MTS_SERVICE: Specifying Service Names for Dispatchers . . . . 4 – 6
MTS_DISPATCHERS: Setting the Initial Number of Dispatchers . . . . 4 – 6
MTS_MAX_ DISPATCHERS: Setting the Maximum Number of Dispatchers . . . . 4 – 7
MTS_SERVERS: Setting the Initial Number of Shared Server Processes . . . . 4 – 7
MTS_MAX_SERVERS: Setting the Maximum Number of Shared Server Processes . . . . 4 – 8
Modifying Server Processes . . . . 4 – 8

Changing the Minimum Number of Shared Server Processes. . . . . 4 – 8
Adding and Removing Dispatcher Processes . . . . 4 – 9
Tracking Oracle7 Processes . . . . 4 – 10
Monitoring the Processes of an Oracle7 Instance . . . . 4 – 10
Trace Files, the ALERT File, and Background Processes . . . . 4 – 12
Starting the Checkpoint Process . . . . 4 – 15
Managing Processes for the Parallel Query Option . . . . 4 – 15
Managing the Query Servers . . . . 4 – 15
Variations in the Number of Query Server Processes . . . . . . 4 – 15
Terminating Sessions . . . . 4 – 16
Identifying Which Session to Terminate . . . . 4 – 17
Terminating an Active Session . . . . 4 – 17
Terminating an Inactive Session . . . . 4 – 18

Chapter 5 **Managing the Online Redo Log . . . . 5 – 1**
Planning the Online Redo Log . . . . 5 – 2
Multiplex the Online Redo Log . . . . 5 – 2
Place Online Redo Log Members on Different Disks . . . . 5 – 3
Set the Size of Online Redo Log Members . . . . 5 – 3
Choose an Appropriate Number of Online Redo Log Files . . . . 5 – 4
Creating Online Redo Log Groups and Members . . . . 5 – 5
Renaming and Relocating Online Redo Log Members . . . . 5 – 6
Dropping Online Redo Log Groups . . . . 5 – 8
Dropping Online Redo Log Members . . . . 5 – 9
Controlling Checkpoints and Log Switches . . . . 5 – 10
Setting Database Checkpoint Intervals . . . . 5 – 11
Forcing a Log Switch . . . . 5 – 12
Forcing a Fast Database Checkpoint Without a Log Switch . . . . 5 – 13
Verifying Blocks in Redo Log Files . . . . 5 – 14
Clearing an Online Redo Log File . . . . 5 – 14
Restrictions . . . . 5 – 14
Listing Information about the Online Redo Log . . . . 5 – 15

Chapter 6 **Managing Control Files . . . . 6 – 1**
Guidelines for Control Files . . . . 6 – 2
Name Control Files . . . . 6 – 2
Multiplex Control Files on Different Disks . . . . 6 – 2
Place Control Files Appropriately . . . . 6 – 3
Manage the Size of Control Files . . . . 6 – 3

Creating Control Files . . . . 6 – 3
Creating Initial Control Files . . . . 6 – 4
Creating Additional Copies of the Control File, and Renaming and Relocating Control Files . . . . 6 – 4
New Control Files . . . . 6 – 5
Creating New Control Files . . . . 6 – 6
Troubleshooting After Creating Control Files . . . . 6 – 8
Checking for Missing or Extra Files . . . . 6 – 8
Handling Errors During CREATE CONTROLFILE . . . . 6 – 9
Dropping Control Files . . . . 6 – 9

Chapter 7 **Managing Job Queues . . . . 7 – 1**
SNP Background Processes . . . . 7 – 2
Multiple SNP Processes . . . . 7 – 2
Starting Up SNP Processes . . . . 7 – 3
Managing Job Queues . . . . 7 – 3
DBMS_JOB Package . . . . 7 – 4
Submitting a Job to the Job Queue . . . . 7 – 4
How Jobs Execute . . . . 7 – 8
Removing a Job From the Job Queue . . . . 7 – 9
Altering a Job . . . . 7 – 10
Broken Jobs . . . . 7 – 11
Forcing a Job to Execute . . . . 7 – 12
Terminating a Job . . . . 7 – 13
Viewing Job Queue Information . . . . 7 – 13

PART III **DATABASE STORAGE**

Chapter 8 **Managing Tablespaces . . . . 8 – 1**
Guidelines for Managing Tablespaces . . . . 8 – 2
Using Multiple Tablespaces . . . . 8 – 2
Specifying Tablespace Storage Parameters . . . . 8 – 2
Assigning Tablespace Quotas to Users . . . . 8 – 3
Creating Tablespaces . . . . 8 – 3
Creating a Temporary Tablespace . . . . 8 – 4
Managing Tablespace Allocation . . . . 8 – 5
Altering Storage Settings for Tablespaces . . . . 8 – 5
Coalescing Free Space . . . . 8 – 6
Altering Tablespace Availability . . . . 8 – 7
Bringing Tablespaces Online . . . . 8 – 7
Taking Tablespaces Offline . . . . 8 – 8

Making a Tablespace Read–Only . . . . 8 – 9
Prerequisites . . . . 8 – 10
Making a Read–Only Tablespace Writeable . . . . 8 – 11
Creating a Read–Only Tablespace on a WORM Device . . . . 8 – 11
Dropping Tablespaces . . . . 8 – 12
Viewing Information About Tablespaces . . . . 8 – 13

Chapter 9 **Managing Datafiles . . . . 9 – 1**
Guidelines for Managing Datafiles . . . . 9 – 2
Number of Datafiles . . . . 9 – 2
Set the Size of Datafiles . . . . 9 – 3
Place Datafiles Appropriately . . . . 9 – 3
Store Datafiles Separately From Redo Log Files . . . . 9 – 3
Creating and Adding Datafiles to a Tablespace . . . . 9 – 4
Changing a Datafile’s Size . . . . 9 – 4
Enabling and Disabling Automatic Extension for a Datafile . . . . 9 – 4
Manually Resizing a Datafile . . . . 9 – 5
Altering Datafile Availability . . . . 9 – 6
Bringing Datafiles Online in ARCHIVELOG Mode . . . . 9 – 7
Taking Datafiles Offline in NOARCHIVELOG Mode . . . . . . 9 – 7
Renaming and Relocating Datafiles . . . . 9 – 7
Renaming and Relocating Datafiles for a Single Tablespace . . . . 9 – 8
Renaming and Relocating Datafiles for Multiple Tablespaces . . . . 9 – 9
Verifying Data Blocks in Datafiles . . . . 9 – 11
Viewing Information About Datafiles . . . . 9 – 12
Listing Status Information About Datafiles: Example . . . . . . 9 – 12

Chapter 10 **Guidelines for Managing Schema Objects . . . . 10 – 1**
Managing Space in Data Blocks . . . . 10 – 2
The PCTFREE Parameter . . . . 10 – 2
The PCTUSED Parameter . . . . 10 – 4
Selecting Associated PCTUSED and PCTFREE Values . . . . . 10 – 6
Setting Storage Parameters . . . . 10 – 7
Storage Parameters You Can Specify . . . . 10 – 7
Setting INITRANS and MAXTRANS . . . . 10 – 10
Setting Default Storage Parameters for Segments in a Tablespace . . . . 10 – 11
Setting Storage Parameters for Data Segments . . . . 10 – 11
Setting Storage Parameters for Index Segments . . . . 10 – 11

Changing Values for Storage Parameters . . . . 10 – 11
Understanding Precedence in Storage Parameters . . . . 10 – 12
Deallocating Space . . . . 10 – 13
Viewing the High Water Mark . . . . 10 – 13
Issuing Space Deallocation Statements . . . . 10 – 14
Understanding Space Use of Datatypes . . . . 10 – 17
Summary of Oracle Datatypes . . . . 10 – 20

Chapter 11 **Managing Tables . . . . 11 – 1**
Guidelines for Managing Tables . . . . 11 – 2
Design Tables Before Creating Them . . . . 11 – 2
Specify How Data Block Space Is to Be Used . . . . 11 – 2
Specify Transaction Entry Parameters . . . . 11 – 3
Specify the Location of Each Table . . . . 11 – 3
Parallelize Table Creation . . . . 11 – 3
Consider Creating UNRECOVERABLE Tables . . . . 11 – 4
Estimate Table Size and Set Storage Parameters . . . . 11 – 4
Plan for Large Tables . . . . 11 – 5
Creating Tables . . . . 11 – 6
Altering Tables . . . . 11 – 7
Manually Allocating Storage for a Table . . . . 11 – 8
Dropping Tables . . . . 11 – 9

Chapter 12 **Managing Views, Sequences and Synonyms . . . . 12 – 1**
Managing Views . . . . 12 – 2
Creating Views . . . . 12 – 2
Modifying a Join View . . . . 12 – 4
Replacing Views . . . . 12 – 8
Dropping Views . . . . 12 – 9
Managing Sequences . . . . 12 – 10
Creating Sequences . . . . 12 – 10
Altering Sequences . . . . 12 – 11
Initialization Parameters Affecting Sequences . . . . 12 – 11
Dropping Sequences . . . . 12 – 11
Managing Synonyms . . . . 12 – 12
Creating Synonyms . . . . 12 – 12
Dropping Synonyms . . . . 12 – 12

Chapter 13 **Managing Indexes** 13 – 1
Guidelines for Managing Indexes 13 – 2
Create Indexes After Inserting Table Data 13 – 3
Limit the Number of Indexes per Table 13 – 3
Specify Transaction Entry Parameters 13 – 4
Specify Index Block Space Use 13 – 4
Specify the Tablespace for Each Index 13 – 4
Parallelize Index Creation 13 – 4
Consider Creating UNRECOVERABLE Indexes 13 – 5
Estimate Index Size and Set Storage Parameters 13 – 5
Considerations Before Disabling or Dropping Constraints 13 – 6
Creating Indexes 13 – 6
Creating an Index Associated with a Constraint 13 – 7
Creating an Index Explicitly 13 – 7
Re–Creating an Existing Index 13 – 7
Altering Indexes 13 – 8
Monitoring Space Use of Indexes 13 – 9
Dropping Indexes 13 – 9

Chapter 14 **Managing Clusters** 14 – 1
Guidelines for Managing Clusters 14 – 2
Cluster Appropriate Tables 14 – 4
Choose Appropriate Columns for the Cluster Key 14 – 4
Specify Data Block Space Use 14 – 4
Specify the Space Required by an Average Cluster Key and Its Associated Rows 14 – 5
Specify the Location of Each Cluster and Cluster Index 14 – 5
Estimate Cluster Size and Set Storage Parameters 14 – 5
Creating Clusters 14 – 6
Creating Clustered Tables 14 – 7
Creating Cluster Indexes 14 – 7
Altering Clusters 14 – 8
Altering Cluster Tables and Cluster Indexes 14 – 9
Dropping Clusters 14 – 9
Dropping Clustered Tables 14 – 10
Dropping Cluster Indexes 14 – 10

Chapter 15 **Managing Hash Clusters** 15 – 1
Guidelines for Managing Hash Clusters 15 – 2
Advantages of Hashing 15 – 2
Disadvantages of Hashing 15 – 3

Estimate Size Required by Hash Clusters and Set Storage Parameters . . . . 15 – 4
Creating Hash Clusters . . . . 15 – 5
Controlling Space Use Within a Hash Cluster . . . . 15 – 5
Altering Hash Clusters . . . . 15 – 8
Dropping Hash Clusters . . . . 15 – 9

Chapter 16 **General Management of Schema Objects . . . . 16 – 1**
Creating Multiple Tables and Views in A Single Operation . . . . 16 – 2
Renaming Schema Objects . . . . 16 – 3
Analyzing Tables, Indexes, and Clusters . . . . 16 – 3
Using Statistics for Tables, Indexes, and Clusters . . . . 16 – 4
Validating Tables, Indexes, and Clusters . . . . 16 – 7
Listing Chained Rows of Tables and Clusters . . . . 16 – 8
Truncating Tables and Clusters . . . . 16 – 9
Enabling and Disabling Triggers . . . . 16 – 11
Enabling Triggers . . . . 16 – 11
Disabling Triggers . . . . 16 – 11
Managing Integrity Constraints . . . . 16 – 12
Managing Constraints That Have Associated Indexes . . . . 16 – 13
Enabling and Disabling Integrity Constraints Upon Definition . . . . 16 – 14
Enabling and Disabling Existing Integrity Constraints . . . . . 16 – 15
Dropping Integrity Constraints . . . . 16 – 15
Reporting Constraint Exceptions . . . . 16 – 16
Managing Object Dependencies . . . . 16 – 18
Manually Recompiling Views . . . . 16 – 19
Manually Recompiling Procedures and Functions . . . . 16 – 19
Manually Recompiling Packages . . . . 16 – 19
Managing Object Name Resolution . . . . 16 – 20
Changing Storage Parameters for the Data Dictionary . . . . 16 – 21
Structures in the Data Dictionary . . . . 16 – 21
Errors that Require Changing Data Dictionary Storage . . . . 16 – 23
Displaying Information About Schema Objects . . . . 16 – 23
Oracle Packages . . . . 16 – 24

Chapter 17 **Managing Rollback Segments . . . . 17 – 1**
Guidelines for Managing Rollback Segments . . . . 17 – 2
Use Multiple Rollback Segments . . . . 17 – 2
Choose Between Public and Private Rollback Segments . . . . 17 – 3
Specify Rollback Segments to Acquire Automatically . . . . . . 17 – 4

Set Rollback Segment Sizes Appropriately . . . . 17 – 4
Create Rollback Segments with Many Equally–Sized Extents . . . . 17 – 5
Set an Optimal Number of Extents for Each Rollback Segment . . . . 17 – 6
Set the Storage Location for Rollback Segments . . . . 17 – 7
Creating Rollback Segments . . . . 17 – 8
Bringing New Rollback Segments Online . . . . 17 – 8
Specifying Storage Parameters for Rollback Segments . . . . 17 – 8
Setting Storage Parameters When Creating a Rollback Segment . . . . 17 – 9
Changing Rollback Segment Storage Parameters . . . . 17 – 9
Altering Rollback Segment Format . . . . 17 – 10
Shrinking a Rollback Segment Manually . . . . 17 – 10
Taking Rollback Segments Online and Offline . . . . 17 – 10
Bringing Rollback Segments Online . . . . 17 – 11
Taking Rollback Segments Offline . . . . 17 – 12
Explicitly Assigning a Transaction to a Rollback Segment . . . . . . 17 – 13
Dropping Rollback Segments . . . . 17 – 14
Monitoring Rollback Segment Information . . . . 17 – 15
Displaying Rollback Segment Information . . . . 17 – 15

PART IV **DATABASE SECURITY**

Chapter 18 **Establishing Security Policies . . . . 18 – 1**
System Security Policy . . . . 18 – 2
Database User Management . . . . 18 – 2
User Authentication . . . . 18 – 2
Operating System Security . . . . 18 – 3
Data Security Policy . . . . 18 – 3
User Security Policy . . . . 18 – 4
General User Security . . . . 18 – 4
End–User Security . . . . 18 – 5
Administrator Security . . . . 18 – 7
Application Developer Security . . . . 18 – 8
Application Administrator Security . . . . 18 – 10
Auditing Policy . . . . 18 – 11

Chapter 19 **Managing Users and Resources . . . . 19 – 1**
Session and User Licensing . . . . 19 – 2
Connecting Privileges . . . . 19 – 2
Setting the Maximum Number of Sessions . . . . 19 – 3
Setting the Session Warning Limit . . . . 19 – 4
Changing Concurrent Usage Limits While the Database is Running . . . . 19 – 4
User Authentication . . . . 19 – 6
Operating System Authentication . . . . 19 – 7
Oracle Users . . . . 19 – 8
Creating Users . . . . 19 – 9
Altering Users . . . . 19 – 12
Dropping Users . . . . 19 – 15
Managing Resources with Profiles . . . . 19 – 16
Creating Profiles . . . . 19 – 16
Assigning Profiles . . . . 19 – 17
Altering Profiles . . . . 19 – 17
Using Composite Limits . . . . 19 – 18
Dropping Profiles . . . . 19 – 19
Enabling and Disabling Resource Limits . . . . 19 – 19
Listing Information About Database Users and Profiles . . . . 19 – 20
Listing Information about Users and Profiles: Examples . . . 19 – 21

Chapter 20 **Managing User Privileges and Roles . . . . 20 – 1**
Identifying User Privileges . . . . 20 – 2
System Privileges . . . . 20 – 2
Object Privileges . . . . 20 – 6
Managing User Roles . . . . 20 – 8
Creating a Role . . . . 20 – 8
Predefined Roles . . . . 20 – 9
Role Authorization . . . . 20 – 9
Dropping Roles . . . . 20 – 11
Granting User Privileges and Roles . . . . 20 – 12
Granting System Privileges and Roles . . . . 20 – 12
Granting Object Privileges and Roles . . . . 20 – 13
Granting Privileges on Columns . . . . 20 – 14
Revoking User Privileges and Roles . . . . 20 – 15
Revoking System Privileges and Roles . . . . 20 – 15
Revoking Object Privileges and Roles . . . . 20 – 15
Effects of Revoking Privileges . . . . 20 – 16
Granting to and Revoking from the User Group PUBLIC . . 20 – 18
Granting Roles Using the Operating System or Network . . . . . . 20 – 18

Using Operating System Role Identification . . . . 20 – 20
Using Operating System Role Management . . . . 20 – 21
Granting and Revoking Roles When OS_ROLES=TRUE . . . 20 – 21
Enabling and Disabling Roles When OS_ROLES=TRUE . . . 20 – 21
Using Network Connections with Operating System Role Management . . . . 20 – 21
Listing Privilege and Role Information . . . . 20 – 22
Listing Privilege and Role Information: Examples . . . . 20 – 23

Chapter 21 **Auditing Database Use . . . . 21 – 1**
Guidelines for Auditing . . . . 21 – 2
Audit via the Database or Operating System . . . . 21 – 2
Keep Audited Information Manageable . . . . 21 – 2
Creating and Deleting the Database Audit Trail Views . . . . 21 – 4
Creating the Audit Trail Views . . . . 21 – 5
Deleting the Audit Trail Views . . . . 21 – 5
Managing Audit Trail Information . . . . 21 – 5
Events Audited by Default . . . . 21 – 6
Setting Auditing Options . . . . 21 – 7
Enabling and Disabling Database Auditing . . . . 21 – 15
Controlling the Growth and Size of the Audit Trail . . . . 21 – 16
Protecting the Audit Trail . . . . 21 – 18
Viewing Database Audit Trail Information . . . . 21 – 18
Listing Active Statement Audit Options . . . . 21 – 21
Listing Active Privilege Audit Options . . . . 21 – 21
Listing Active Object Audit Options for Specific Objects . . . 21 – 21
Listing Default Object Audit Options . . . . 21 – 22
Listing Audit Records . . . . 21 – 22
Listing Audit Records for the AUDIT SESSION Option . . . . 21 – 22
Auditing Through Database Triggers . . . . 21 – 22

PART V **DATABASE BACKUP AND RECOVERY**

Chapter 22 **Archiving Redo Information . . . . 22 – 1**
Choosing Between NOARCHIVELOG and ARCHIVELOG Mode . . . . 22 – 2
Running a Database in NOARCHIVELOG Mode . . . . 22 – 2
Running a Database in ARCHIVELOG Mode . . . . 22 – 2
Turning Archiving On and Off . . . . 22 – 4
Setting the Initial Database Archiving Mode . . . . 22 – 4
Changing the Database Archiving Mode . . . . 22 – 5

Enabling Automatic Archiving . . . . 22 – 6
Disabling Automatic Archiving . . . . 22 – 7
Performing Manual Archiving . . . . 22 – 8
Tuning Archiving . . . . 22 – 9
Minimizing the Impact on System Performance . . . . 22 – 9
Improving Archiving Speed . . . . 22 – 10
Displaying Archiving Status Information . . . . 22 – 10
Specifying the Archived Redo Log Filename Format and Destination . . . . 22 – 11

Chapter 23 **Backing Up a Database . . . . 23 – 1**
Guidelines for Database Backups . . . . 23 – 2
Test Backup and Recovery Strategies . . . . 23 – 2
Perform Operating System Backups Frequently and Regularly . . . . 23 – 2
Backup Appropriate Portions of the Database When Making Structural Changes . . . . 23 – 3
Back Up Often–Used Tablespaces Frequently . . . . 23 – 3
Keep Older Backups . . . . 23 – 3
Export Database Data for Added Protection and Flexibility . . . . 23 – 4
Consider Distributed Database Backups . . . . 23 – 4
Back Up after Creating Unrecoverable Objects . . . . 23 – 4
Creating a Backup Strategy . . . . 23 – 5
Backup Strategies in NOARCHIVELOG Mode . . . . 23 – 5
Backup Strategies in ARCHIVELOG Mode . . . . 23 – 6
Read–Only Tablespaces and Backup . . . . 23 – 7
Performing Backups . . . . 23 – 7
Listing Database Files Before Backup . . . . 23 – 8
Performing Full Offline Backups . . . . 23 – 8
Performing Partial Backups . . . . 23 – 10
Performing Control File Backups . . . . 23 – 15
Recovering From an Incomplete Online Tablespace Backup . . . . 23 – 17
Using the Export and Import Utilities for Supplemental Database Protection . . . . 23 – 18
Using Export . . . . 23 – 18
Using Import . . . . 23 – 19

Chapter 24 **Recovering a Database . . . . 24 – 1**
Fundamental Recovery Concepts and Strategies . . . . 24 – 2
Important Recovery Data Structures . . . . 24 – 2
Recovery Operations . . . . 24 – 3
Recovery Planning and Strategies . . . . 24 – 3
Preparing for Media Recovery . . . . 24 – 7
Media Recovery Commands . . . . 24 – 7
Issues Common to All Media Recovery Operations . . . . 24 – 8
Restoring a Full Backup, NOARCHIVELOG Mode . . . . 24 – 16
Specifying Parallel Recovery . . . . 24 – 17
Performing Complete Media Recovery . . . . 24 – 18
Performing Closed Database Recovery . . . . 24 – 18
Performing Open–Database, Offline–Tablespace Recovery . . . . 24 – 20
Performing Open–Database, Offline–Tablespace Individual Recovery . . . . 24 – 22
Performing Incomplete Media Recovery . . . . 24 – 24
Performing Cancel–Based Recovery . . . . 24 – 24
Performing Time–Based Recovery . . . . 24 – 28
Performing Change–Based Recovery . . . . 24 – 33
Preparing for Disaster Recovery . . . . 24 – 38
Planning and Creating a Standby Database . . . . 24 – 38
Altering the Physical Structure of the Primary Database . . . . 24 – 41
Unrecoverable Objects and Recovery . . . . 24 – 46
Read–Only Tablespaces and Recovery . . . . 24 – 46
Using a Backup Control File . . . . 24 – 46
Re–Creating a Control File . . . . 24 – 47
Examples of Recovery Procedures . . . . 24 – 47
Types of Media Failures . . . . 24 – 47
Loss of Datafiles . . . . 24 – 48
Loss of Online Redo Log Files . . . . 24 – 49
Loss of Archived Redo Log Files . . . . 24 – 52
Loss of Control Files . . . . 24 – 53
Recovery From User Errors . . . . 24 – 55

PART VI **REFERENCE**

Appendix A **Space Estimations for Schema Objects . . . . A – 1**

Index

PART

I

Basic Database Administration

CHAPTER

1

The Oracle7 Database Administrator

This chapter describes the responsibilities of the person who administers the Oracle7 Server, the database administrator.

The following topics are included:

- Types of Oracle7 Users
- Database Administrator Security and Privileges
- Database Administrator Authentication
- Password File Administration
- Database Administrator Utilities
- Initial Priorities of a Database Administrator
- Identifying Oracle Software Releases

Types of Oracle7 Users

At your site, the types of users and their responsibilities may vary. For example, at a large site the duties of a database administrator might be divided among several people.

This section includes the following topics:

- Database Administrators
- Security Officers
- Application Developers
- Application Administrators
- Database Users
- Network Administrators

Database Administrators

Because an Oracle7 database system can be quite large and have many users, someone or some group of people must manage this system. The *database administrator* (DBA) is this manager. Every database requires at least one person to perform administrative duties.

A database administrator's responsibilities can include the following tasks:

- installing and upgrading the Oracle7 Server and application tools
- allocating system storage and planning future storage requirements for the database system
- creating primary database storage structures (tablespaces) after application developers have designed an application
- creating primary objects (tables, views, indexes) once application developers have designed an application
- modifying the database structure, as necessary, from information given by application developers
- enrolling users and maintaining system security
- ensuring compliance with your Oracle7 license agreement
- controlling and monitoring user access to the database
- monitoring and optimizing the performance of the database

- planning for backup and recovery of database information
- maintaining archived data on tape
- backing up and restoring the database
- contacting Oracle Corporation for technical support

Security Officers

In some cases, a database might also have one or more security officers. A *security officer* is primarily concerned with enrolling users, controlling and monitoring user access to the database, and maintaining system security. You might not be responsible for these duties if your site has a separate security officer.

Application Developers

An *application developer* designs and implements database applications An application developer's responsibilities include the following tasks:

- designing and developing the database application
- designing the database structure for an application
- estimating storage requirements for an application
- specifying modifications of the database structure for an application
- relaying the above information to a database administrator
- tuning the application during development
- establishing an application's security measures during development

Application Administrators

An Oracle site might also have one or more application administrators. An *application administrator* is responsible for the administration needs of a particular application.

Database Users

Database users interact with the database via applications or utilities. A typical user's responsibilities include the following tasks:

- entering, modifying, and deleting data, where permitted
- generating reports of data

Network Administrators

At some sites there may be one or more network administrators. Network administrators may be responsible for administering Oracle7 networking products, such as SQL*Net.

See Also: "Network Administration" in *Oracle7 Server Distributed Systems, Volume I.*

Database Administrator Security and Privileges

To accomplish administrative tasks in Oracle7, you need extra privileges both within the database and possibly in the operating system of the server on which the database runs. Access to a database administrator's account should be tightly controlled.

This section includes the following topics:

- The Database Administrator's Operating System Account
- Database Administrator Usernames
- The DBA Role

See Also: "Administrator Security" on page 18 – 7.

The Database Administrator's Operating System Account

To perform many of the administrative duties for a database, you must be able to execute operating system commands. Depending on the operating system that executes Oracle7, you might need an operating system account or ID to gain access to the operating system. If so, your operating system account might require more operating system privileges or access rights than many database users require (for example, to perform Oracle7 software installation). Although you do not need the Oracle7 files to be stored in your account, you should have access to them.

In addition, the Server Manager program requires that your operating system account or ID be distinguished in some way to allow you to use *operating system privileged* Server Manager commands.

See Also: The method of distinguishing a database administrator's account is operating system–specific. See your operating system–specific Oracle documentation for information.

Database Administrator Usernames

Two user accounts are automatically created with the database and granted the DBA role. These two user accounts are:

- SYS (initial password: CHANGE_ON_INSTALL)
- SYSTEM (initial password: MANAGER)

These two usernames are described in the following sections.

> **Note:** To prevent inappropriate access to the data dictionary tables, you must change the passwords for the SYS and SYSTEM usernames immediately after creating an Oracle7 database.

You will probably want to create at least one additional administrator username to use when performing daily administrative tasks.

SYS

When any database is created, the user SYS, identified by the password CHANGE_ON_INSTALL, is automatically created and granted the DBA role.

All of the base tables and views for the database's data dictionary are stored in the schema SYS. These base tables and views are critical for the operation of Oracle7. To maintain the integrity of the data dictionary, tables in the SYS schema are manipulated only by Oracle7; they should never be modified by any user or database administrator, and no one should create any tables in the schema of the user SYS. (However, you can change the storage parameters of the data dictionary settings if necessary.)

Most database users should never be able to connect using the SYS account. You can connect to the database using this account but should do so only when instructed by Oracle personnel or documentation.

SYSTEM

Also when a database is created, the user SYSTEM, identified by the password MANAGER, is automatically created and granted all system privileges for the database.

The SYSTEM username creates additional tables and views that display administrative information, and internal tables and views used by Oracle tools. Never create tables of interest to individual users in the SYSTEM schema.

See Also: "Altering Users" on page 19 – 12.

"Changing Storage Parameters for the Data Dictionary" on page 16 – 21.

"Administrator Security" on page 18 – 7.

The DBA Role

A predefined role, named "DBA", is automatically created with every Oracle7 database. This role contains all database system privileges. Therefore, it is very powerful and should only be granted to fully functional database administrators.

Database Administrator Authentication

Database administrators must often perform special operations such as shutting down or starting up a database. Because these operations should not be performed by normal database users, the database administrator usernames need a more secure authentication scheme.

This section includes the following topics:

- Selecting an Authentication Method
- Using Operating System Authentication
- OSOPER and OSDBA
- Using a Password File

Selecting an Authentication Method

The following methods for authenticating database administrators replace the CONNECT INTERNAL syntax provided with earlier versions of the Oracle7 Server (CONNECT INTERNAL continues to be supported for backwards compatibility only):

- operating system authentication
- password files

Depending on whether you wish to administer your database locally on the same machine that the database resides or if you wish to administer many different databases from a single remote client, you can choose between operating system authentication or password files to authenticate database administrators. Figure 1 – 1 illustrates the choices you have for database administrator authentication schemes.

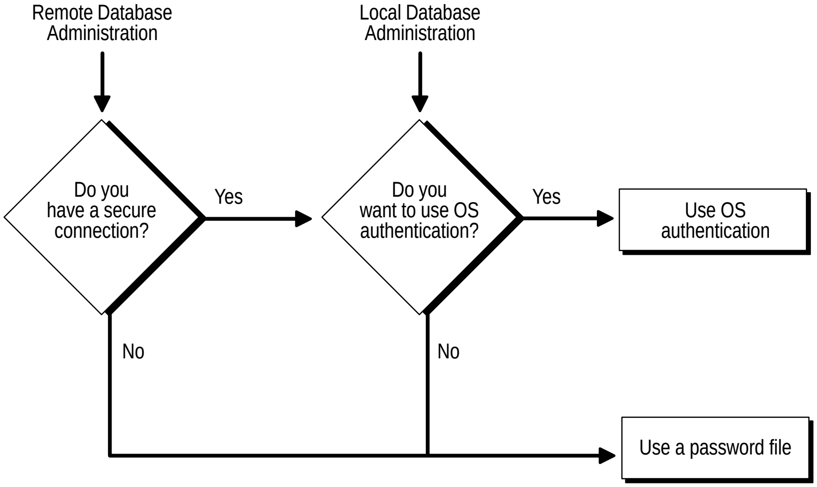

Remote Database Administration
Local Database Administration
Do you have a secure connection?
Yes
Do you want to use OS authentication?
Yes
Use OS authentication
No
No
Use a password file

Figure 1 – 1 Database Administrator Authentication Methods

On most operating systems, OS authentication for database administrators involves placing the OS username of the database administrator in a special group (on UNIX systems, this is the DBA group) or giving that OS username a special process right.

The database uses password files to keep track of database usernames that have been granted administrator privileges.

See Also: "User Authentication" in *Oracle7 Server Concepts*.

Using Operating System Authentication

If you choose, you can have your operating system authenticate users performing database administration operations.

To Use Operating System Authentication

1. Set up the user to be authenticated by the operating system.
2. Make sure that the initialization parameter, REMOTE_LOGIN_PASSWORD, is set to NONE, which is the default value for this parameter.
3. Authenticated users should now be able to connect to a local database, or to connect to a remote database over a secure connection, by typing one of the following commands:

```
CONNECT / AS SYSOPER
CONNECT / AS SYSDBA
```

If you successfully connect as INTERNAL using an earlier release of Oracle7, you should be able to continue to connect successfully using the new syntax shown in step 3.

> **Note:** Note that to connect as SYSOPER or SYSDBA using OS authentication you do not have to have been granted the SYSOPER or SYSDBA system privileges. Instead, the server verifies that you have been granted the appropriate OSDBA or OSOPER roles at the operating system level.

See Also: "Operating System Authentication" on page 19 – 7.

OSOPER and OSDBA

Two special operating system roles control database administrator logins when using operating system authentication: OSOPER and OSDBA.

OSOPER Permits the user to perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN / MOUNT, ALTER DATABASE BACKUP,

	ARCHIVE LOG, and RECOVER, and includes the RESTRICTED SESSION privilege.
OSDBA	Contains all system privileges with ADMIN OPTION, and the OSOPER role; permits CREATE DATABASE and time–based recovery.

OSOPER and OSDBA can have different names and functionality, depending on your operating system.

The OSOPER and OSDBA roles can only be granted to a user through the operating system. They cannot be granted through a GRANT statement, nor can they be revoked or dropped. When a user logs on with administrator privileges and REMOTE_LOGIN_PASSWORDFILE is set to NONE, Oracle7 communicates with the operating system and attempts to enable first OSDBA and then, if unsuccessful, OSOPER. If both attempts fail, the connection fails. How you grant these privileges through the operating system is operating system–specific.

If you are performing remote database administration, you should consult your SQL*Net documentation to determine if you are using a secure connection. Most popular connection protocols, such as TCP/IP and DECnet, are not secure, regardless of which version of SQL*Net you are using.

See Also: For information about OS authentication of database administrators, see your operating system–specific Oracle documentation.

Using an Authentication Password File

If you have determined that you need to use a password file to authenticate users performing database administration, you must complete the steps outlined below. Each of these steps is explained in more detail in the following sections of this chapter.

To Use a Password File to Authenticate Users

1. Create the password file using the ORAPWD utility.

   ```
   ORAPWD FILE=filename PASSWORD=password ENTRIES=max_users
   ```

2. Set the REMOTE_LOGIN_PASSWORDFILE initialization parameter to EXCLUSIVE.
3. Add users to the password file by using SQL to grant the appropriate privileges to each user who needs to perform database administration, as shown in the following examples.

   ```
   GRANT SYSDBA TO scott
   GRANT SYSOPER TO scott
   ```

The privilege SYSDBA permits the user to perform the same operations as OSDBA. Likewise, the privilege SYSOPER permits the user to perform the same operations as OSOPER.

4. Privileged users should now be able to connect to the database by using a command similar to the one shown below.

   ```
   CONNECT scott/tiger@acct.hq.com AS SYSDBA
   ```

See Also: “OSOPER and OSDBA” on page 1 – 7.

Some platforms provided support for password files before release 7.1. If you are currently using such a password file, you should consult your operating system–specific Oracle documentation for additional information on migrating to the new password file utility.

Password File Administration

You can create a password file using the password file creation utility, ORAPWD or, for selected operating systems, you can create this file as part of your standard installation.

This section includes the following topics:

- Using ORAPWD
- Setting REMOTE_LOGIN_PASSWORDFILE
- Adding Users to a Password File
- Connecting with Administrator Privileges
- Maintaining a Password File

See Also: See your operating system–specific Oracle documentation for information on using the installer utility to install the password file.

Using ORAPWD

When you invoke the password file creation utility without supplying any parameters, you receive a message indicating the proper use of the command as shown in the following sample output:

```
> orapwd
Usage: orapwd file=<fname> password=<password> entries=<users>

  where
    file – name of password file (mand),
    password – password for SYS and INTERNAL (mand),
    entries – maximum number of distinct DBAs and OPERs (opt),
  There are no spaces around the equal–to (=) character.
```

For example, the following command creates a password file named ACCT.PWD that allows up to 30 privileged users with different passwords. The file is initially created with the password SECRET for users connecting as INTERNAL or SYS:

```
> ORAPWD FILE=acct.pwd PASSWORD=secret ENTRIES=30
```

Following are descriptions of the parameters in the ORAPWD utility.

FILE

This parameter sets the name of the password file being created. You must specify the full pathname for the file. The contents of this file are encrypted, and the file is not user–readable. This parameter is mandatory.

On some platforms, the name of the password file is derived from the *system identifier* (SID). If so, you must specify this predefined name when using the ORAPWD utility. On other systems the name of the password file is stored in an environment variable, such as ORA_*sid*_PWFILE. If your system uses an environment variable, you must set this variable on the server platform to match the pathname that you specified for the file before starting the instance.

If you are running multiple instances of Oracle7 using the Oracle7 Parallel Server, the environment variable for each instance should point to the same password file.

⚠ **Warning:** It is critically important to the security of your system that you protect your password file, and environment variables that identify the location of the password file. Any user with access to these could potentially compromise the security of the connection.

PASSWORD

This parameter sets the password for INTERNAL and SYS. If you issue the ALTER USER command to change the password after connecting to the database, both the password stored in the data dictionary and the password stored in the password file are updated. The INTERNAL user is supported for backwards compatibility only. This parameter is mandatory.

ENTRIES

This parameter sets the maximum number of entries allowed in the password file. This corresponds to the maximum number of distinct users allowed to connect to the database as SYSDBA or SYSOPER. Entries can be reused as users are added to and removed from the password file. This parameter is required if you ever want this password file to be EXCLUSIVE.

Warning: If you ever need to exceed this limit, you must create a new password file. It is safest to select a number larger than you think you will ever need.

See Also: Consult your operating system–specific Oracle documentation for the exact name of the password file, or for the name of the environment variable used to specify this name for your operating system.

Setting REMOTE_LOGIN_PASSWORDFILE

In addition to creating the password file, you must also set the initialization parameter REMOTE_LOGIN_PASSWORDFILE to the appropriate value. The values recognized are described below.

Note: To STARTUP an instance or database, you must use Server Manager. You must specify a database name and a parameter file to initialize the instance settings. You may specify a fully–qualified remote database name using SQL*Net. However, the initialization parameter file and any associated files, such as a configuration file, must exist on the client machine. That is, the parameter file must be on the machine where you are running Server Manager.

NONE

Setting this parameter to NONE causes Oracle7 to behave as if the password file does not exist. That is, no privileged connections are allowed over non–secure connections. NONE is the default value for this parameter.

EXCLUSIVE

An EXCLUSIVE password file can be used with only one database. Only an EXCLUSIVE file can contain the names of users other than SYS and INTERNAL. Using an EXCLUSIVE password file allows you to grant SYSDBA and SYSOPER system privileges to individual users and have them connect as themselves.

SHARED

A SHARED password file can be used by multiple databases. However, the only users recognized by a SHARED password file are SYS and INTERNAL; you cannot add users to a SHARED password file. All users needing SYSDBA or SYSOPER system privileges must connect using the same name, SYS, and password. This option is useful if you have a single DBA administering multiple databases.

Suggestion: To achieve the greatest level of security, you should set the REMOTE_LOGIN_PASSWORDFILE file initialization parameter to EXCLUSIVE immediately after creating the password file.

Adding Users to a Password File

When you grant SYSDBA or SYSOPER privileges to a user, that user's name and privilege information is added to the password file. If the server does not have an EXCLUSIVE password file, that is, if the initialization parameter REMOTE_LOGIN_PASSWORDFILE is NONE or SHARED, you receive an error message if you attempt to grant these privileges.

A user's name only remains in the password file while that user has at least one of these two privileges. When you revoke the last of these privileges from a user, that user is removed from the password file.

To Create a Password File and Add New Users to It

1. Follow the instructions on page 1 – 9 for creating a password file.
2. Set the REMOTE_LOGIN_PASSWORDFILE initialization parameter to EXCLUSIVE.
3. Connect with SYSDBA privileges as shown in the following example:

   ```
   CONNECT SYS/change_on_install AS SYSDBA
   ```

4. Startup the instance and create the database if necessary, or mount and open an existing database.
5. Create users as necessary. Grant SYSOPER or SYSDBA privileges to yourself and other users as appropriate.
6. These users are now added to the password file and can connect to the database as SYSOPER or SYSDBA with a username and password (instead of using SYS). The use of a password file does not prevent OS authenticated users from connecting if they meet the criteria for OS authentication.

Granting and Revoking SYSOPER and SYSDBA Privileges

If your server is using an EXCLUSIVE password file, use the GRANT command to grant the SYSDBA or SYSOPER system privilege to a user, as shown in the following example:

```
GRANT SYSDBA TO scott
```

Use the REVOKE command to revoke the SYSDBA or SYSOPER system privilege from a user, as shown in the following example:

```
REVOKE SYSDBA FROM scott
```

Because SYSDBA and SYSOPER are the most powerful database privileges, the ADMIN OPTION is not used. Only users currently connected as SYSDBA (or INTERNAL) can grant SYSDBA or SYSOPER system privileges to another user. This is also true of REVOKE. These privileges cannot be granted to roles, since roles are only available after database startup. Do not confuse the SYSDBA and SYSOPER database privileges with operating system roles, which are a completely independent feature.

Listing Password File Members

Use the V$PWFILE_USERS view to determine which users have been granted SYSDBA and SYSOPER system privileges for a database. The columns displayed by this view are as follows:

USERNAME The name of the user that is recognized by the password file.

SYSDBA If the value of this column is TRUE, the user can log on with SYSDBA system privileges.

SYSOPER If the value of this column is TRUE, the user can log on with SYSOPER system privileges.

Connecting with Administrator Privileges

When you connect with SYSOPER or SYSDBA privileges using a username and password, you are connecting with a default schema of SYS, not the schema that is generally associated with your username.

Use the AS SYSDBA or AS SYSOPER clauses of the Server Manager CONNECT command to connect with administrator privileges.

Connecting with Administrator Privileges: Example

For example, assume user SCOTT has issued the following commands:

```
CONNECT scott/tiger
CREATE TABLE scott_test(name VARCHAR2(20));
```

Later, when SCOTT issues these commands:

```
CONNECT scott/tiger AS SYSDBA
SELECT * FROM scott_test;
```

He receives an error that SCOTT_TEST does not exist. That is because SCOTT now references the SYS schema by default, whereas the table was created in the SCOTT schema.

Non–Secure Remote Connections

To connect to Oracle7 as a privileged user over a non–secure connection, you must meet the following conditions:

- The server to which you are connecting must have a password file.
- You must be granted the SYSOPER or SYSDBA system privilege.
- You must connect using a username and password.

Local and Secure Remote Connections

To connect to Oracle7 as a privileged user over a local or a secure remote connection, you must meet either of the following sets of conditions:

- You can connect using a password file, provided that you meet the criteria outlined for non–secure connections in the previous bulleted list.
- If the server is not using a password file, or you have not been granted SYSOPER or SYSDBA privileges and are therefore not in the password file, your operating system name must be authenticated for a privileged connection by the operating system. This form of authentication is operating system–specific.

Consult your operating system–specific Oracle documentation for details on operating system authentication.

See Also: "Password File Administration" on page 1 – 9.

Maintaining a Password File

This section describes how to expand, relocate, and remove the password file, as well as how to avoid changing the state of the password file.

Expanding the Number of Password File Users

If you receive the file full error (ORA–1996) when you try to grant SYSDBA or SYSOPER system privileges to a user, you must create a larger password file and re–grant the privileges to the users.

To Replace a Password File

1. Note which users have SYSDBA or SYSOPER privileges by querying the V$PWFILE_USERS view.
2. Shut down the database.
3. Delete the existing password file.
4. Follow the instructions for creating a new password file using the ORAPWD utility on page 1 – 9. Be sure to set the ENTRIES parameter to a sufficiently large number.
5. Follow the instructions for adding users to the password file on page 1 – 12.

Relocating the Password File

After you have created the password file, you can relocate it as you choose. After relocating the password file, you must reset the appropriate environment variables to the new pathname. If your operating system uses a predefined pathname, you cannot change the password file location.

Removing a Password File

If you determine that you no longer need to use a password file to authenticate users, you can delete the password file and reset the REMOTE_LOGIN_PASSWORDFILE initialization parameter to NONE. After removing this file, only users who can be authenticated by the operating system can perform database administration operations.

⚠ **Warning:** Do not remove or modify the password file if you have a database or instance mounted using REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE (or SHARED). If you do, you will be unable to reconnect remotely using the password file. Even if you replace it, you cannot use the new password file, because the timestamp and checksums will be wrong.

Changing the Password File State

The password file state is stored in the password file. When you first create a password file, its default state is SHARED. You can change the state of the password file by setting the parameter REMOTE_LOGIN_PASSWORDFILE. When you STARTUP an instance, Oracle7 retrieves the value of this parameter from the initialization parameter file stored on your client machine. When you mount the database, Oracle7 compares the value of this parameter to the value stored in the password file. If these values do not match, the value stored in the file is overwritten.

> **Warning:** You should use caution to ensure that an EXCLUSIVE password file is not accidentally changed to SHARED. If you plan to allow instance STARTUP from multiple clients, each of those clients must have an initialization parameter file, and the value of the parameter REMOTE_LOGIN_PASSWORDFILE must be the same in each of these files. Otherwise, the state of the password file could change depending upon where the instance was started.

Database Administrator Utilities

Several utilities are available to help you maintain and control the Oracle7 Server.

The following topics are included in this section:

- Server Manager
- SQL*Loader
- Export and Import

Server Manager

Server Manager allows you to monitor and control an Oracle7 database. All administrative operations discussed in this book are executed using Server Manager. Server Manager has both GUI (Graphical User Interface) and line mode interfaces.

Server Manager uses a superset of ANSI/ISO standard SQL commands. The most common administrative commands are available in the menus of Server Manager/GUI. Commands used less frequently can be typed into a Server Manager SQL Worksheet and executed.

See Also: *Oracle Server Manager User's Guide.*

SQL*Loader

SQL*Loader is used by both database administrators and users of Oracle7. It loads data from standard operating system files (files in text or C data format) into Oracle7 database tables.

See Also: *Oracle7 Server Utilities.*

Export and Import

The Export and Import utilities allow you to move existing data in Oracle7 format to and from Oracle7 databases. For example, export files can archive database data, or move data among different Oracle7 databases that run on the same or different operating systems.

See Also: *Oracle7 Server Utilities.*

Initial Priorities of a Database Administrator

In general, you must perform a series of steps to get the database system up and running, and then maintain it.

The following steps are required to configure an Oracle7 Server and database on any type of computer system. The following sections include details about each step.

To Configure an Oracle7 Server

- Step 1: Install the Oracle7 Software
- Step 2: Evaluate the Database Server Hardware
- Step 3: Plan the Database
- Step 4: Create and Open the Database
- Step 5: Implement the Database Design
- Step 6: Back up the Database
- Step 7: Enroll System Users
- Step 8: Tune Database Performance

> **Note:** If migrating to a new release, back up your existing production database before installation. For more information on preserving your existing production database, see Chapter 1 of the *Oracle7 Server Migration*.

Step 1: Install the Oracle7 Software

As the database administrator, you must install the Oracle7 Server software and any front–end tools and database applications that access the database. In some distributed processing installations, the database is controlled by a central computer and the database tools and applications are executed on remote machines; in this case, you must also install the Oracle7 SQL*Net drivers necessary to connect the remote machines to the computer that executes Oracle7.

See Also: "Identifying Oracle Software Releases" on page 1 – 20.

For specific requirements and instructions for installation, see your operating system–specific Oracle documentation and your installation guides for your front–end tools and SQL*Net drivers.

Step 2: Evaluate the Database Server Hardware

After installation, evaluate how Oracle7 and its applications can best use the available computer resources. This evaluation should reveal the following information:

- how many disk drives are available to Oracle7 and its databases
- how many, if any, dedicated tape drives are available to Oracle7 and its databases
- how much memory is available to the instances of Oracle7 you will run (See your system's configuration documentation)

Step 3: Plan the Database

As the database administrator, you must plan:

- the database's logical storage structure
- the overall database design
- a backup strategy for the database

It is important to plan how the logical storage structure of the database will affect system performance and various database management operations. For example, before creating any tablespaces for your database, you should know how many data files will make up the tablespace, where the data files will be physically stored (on which disk drives), and what type of information will be stored in each tablespace. When planning the database's overall logical storage structure, take into account the effects that this structure will have when the database is actually created and running. Such considerations include how the database's logical storage structure will affect the following items:

- the performance of the computer executing Oracle7
- the performance of the database during data access operations
- the efficiency of backup and recovery procedures for the database

Plan the relational design of the database's objects and the storage characteristics for each of these objects. By planning relationships between objects and the physical storage of each object before creating it, you can directly impact the performance of the database as a unit. Be sure to plan for the growth of the database.

In distributed database environments, this planning stage is extremely important. The physical location of highly accessed data can dramatically affect application performance.

During the above planning phases, also plan a backup strategy for the database. After developing this strategy, you might find that you want to alter the database's planned logical storage structure or database design to improve backup efficiency.

It is beyond the scope of this book to discuss relational and distributed database design; if you are not familiar with such design issues, refer to accepted industry–standard books that explain these studies. See Chapters 9 through 17 for specific information on creating logical storage structures, objects, and integrity constraints for your database.

Step 4: Create and Open the Database

Once you have finalized the database design, you can create the database and open it for normal use. Depending on your operating system, a database may already have been created during the installation procedure for Oracle7. If so, all you need to do is start an instance, and mount and open the initial database.

To determine if your operating system creates an initial database during the installation of Oracle7, check your installation or user's guide. If no database is created during installation or you want to create an additional database, see Chapter 2 for this procedure. See Chapter 3 for database and instance startup and shutdown procedures.

Step 5: Implement the Database Design

Once you have created and started the database, you can create the database's planned logical structure by creating all necessary rollback segments and tablespaces. Once this is built, you can create the objects for your database.

See Chapters 8 through 17 for instructions on creating logical storage structures and objects for your database.

Step 6: Back up the Database

After you have created the database structure, carry out the planned backup strategy for your database by creating any additional redo log files, taking the first full database backup (online or offline), and scheduling future database backups at regular intervals.

See Chapters 22 through 24 for instructions on customizing your backup operations and performing recovery procedures.

Step 7: Enroll System Users

Once you have backed up the database structure, you can begin to enroll the users of the database in accordance with your Oracle7 license agreement, create roles for these users, and grant appropriate roles to them.

See Chapters 18 through 20 for the procedures to create user accounts and roles, and information on complying with your license agreement.

Step 8: Tune Database Performance

Optimizing the database system's performance is one of your ongoing responsibilities.

See Also: "Initial Tuning Guidelines" on page 2 – 14 describes steps you can take to start tuning your database immediately after creation.

Oracle7 Server Tuning manual, for information about tuning your database and applications.

Identifying Oracle Software Releases

Because Oracle products are always undergoing development and change, several releases of the products can be in use at any one time. To identify a software product fully, as many as five numbers may be required.

This section includes the following topics:

- Release Number Format
- Versions of Other Oracle Software
- Checking Your Current Release Number

Release Number Format

An Oracle7 Server distribution tape might be labeled "Release 7.0.4.1." The following sections translate this number.

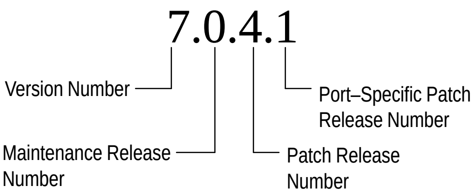

7.0.4.1
Version Number
Port–Specific Patch Release Number
Maintenance Release Number
Patch Release Number

Figure 1 – 2 Example of an Oracle7 Release Number

Version Number

The version number, such as **7**, is the most general identifier. A *version* is a major new edition of the software, which usually contains significant new functionality.

Maintenance Release Number

The maintenance release number signifies different releases of the general version, starting with 0, as in version 7.**0**. The maintenance release number increases when bug fixes or new features to existing programs become available.

Patch Release Number

The patch release number identifies a specific level of the object code, such as 7.0.**4**. A patch release contains fixes for serious bugs that cannot wait until the next maintenance release. The first distribution of a maintenance release always has a patch number of 0.

Port–Specific Patch Release Number

A fourth number (and sometimes a fifth number) can be used to identify a particular emergency patch release of a software product on that operating system, such as 7.0.4.**1**. or 7.0.4.**1.3**. An emergency patch is not usually intended for wide distribution; it usually fixes or works around a particular, critical problem.

Examples of Release Numbers

The following examples show possible release numbers for Oracle7:

7.0.0	the first distribution of Oracle7
7.1.0	the first maintenance release of Oracle7
7.2.0	the second maintenance release (the third release in all) of Oracle7
7.2.2	the second patch release after the second maintenance release

Versions of Other Oracle Software

As Oracle Corporation introduces new products and enhances existing ones, the version numbers of the individual products increment independently. Thus, you might have an Oracle7 Server Release 7.0.12.2 system working with Oracle Forms Version 4.0.3, SQL*Plus Version 3.1.9, and Pro*FORTRAN Version 1.5.2. (These numbers are used only for illustration.)

Checking Your Current Release Number

To see which release of Oracle and its components you are using, query the data dictionary view PRODUCT_COMPONENT_VERSION, as shown below (This information is useful if you need to call Oracle Support.):

```
SVRMGR> SELECT * FROM product_component_version;
PRODUCT                     VERSION             STATUS
-------------------------- ------------------- ------------
CORE                        3.4.1.0.0           Production
NLSRTL                      3.1.3.0.0           Production
```

```
Oracle7 Server             7.2.1.0.0           Beta Release
PL/SQL                     2.2.1.0.0           Beta
TNS for SunOS:             2.1.4.0.0           Production
5 rows selected.
```

CHAPTER

2 Creating a Database

This chapter lists the steps necessary to create an Oracle7 database, and includes the following topics:

- Considerations Before Creating a Database
- Creating an Oracle7 Database
- Parameters
- Considerations After Creating a Database
- Initial Tuning Guidelines

See Also: *Trusted Oracle7 Server Administrator's Guide.*

Oracle Server Manager User's Guide.

Considerations Before Creating a Database

This section includes the following topics:

- Creation Prerequisites
- Using an Initial Database
- Migrating an Older Version of the Database

Database creation prepares several operating system files so they can work together as an Oracle7 database. You need only create a database once, regardless of how many datafiles it has or how many instances access it. Creating a database can also erase information in an existing database and create a new database with the same name and physical structure.

Creating a database includes the following operations:

- creating new datafiles or erasing data that existed in previous datafiles
- creating structures that Oracle7 requires to access and use the database (the data dictionary)
- creating and initializing the control files and redo log files for the database

Consider the following issues before you create a database:

- Plan your database tables and indexes, and estimate how much space they will require.

  For information about tables, indexes, and space management, see Chapters 9 through 16.

- Plan how to protect your new database, including the configuration of its online and archived redo log (and how much space it will require), and a backup strategy.

  For information about the online and archive redo logs, see Chapters 5 and 22 respectively.

  For information about database backup and recovery, see Chapters 23 and 24.

- Select the database character set. You must specify the database character set when you create the database. If you create the database with the wrong character set by mistake, you can update the SYS.PROPS$ table with the new character set; however, you cannot change characters already there. All character data, including data in the data dictionary, is stored in the database character set. If users access the database using a different character set, the database character set should be the same as, or a superset of, all character sets they use.

Also become familiar with the principles and options of starting up and shutting down an instance, mounting and opening a database, and using parameter files. For information about starting up and shutting down, see Chapter 3.

See Also: "National Language Support" in the *Oracle7 Server Reference*.

Creation Prerequisites

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

- the operating system privileges associated with a fully operational database administrator
- sufficient memory to start the Oracle7 instance
- sufficient disk storage space for the planned database on the computer that executes Oracle7

Using an Initial Database

Depending on your operating system, a database might have been created automatically as part of the installation procedure for Oracle7. You can use this initial database and customize it to meet your information management requirements, or discard it and create one or more new databases to replace it.

Migrating an Older Version of the Database

If you are using a previous release of Oracle7, database creation is required only if you want an entirely new database. Otherwise, you can migrate your existing Oracle7 databases managed by a previous version of Oracle7 and use them with the new version of the Oracle7 software.

See Also: *Oracle7 Server Migration* manual for information about migrating an existing database.

For information about migrating to Trusted Oracle7, see the *Trusted Oracle7 Server Administrator's Guide.*

For more information about migrating an existing database, see your operating system–specific Oracle documentation.

Creating an Oracle7 Database

This section includes the following topics:

- Steps for Creating an Oracle7 Database
- Creating a Database: Example
- Troubleshooting Database Creation
- Dropping a Database

Steps for Creating an Oracle7 Database

These steps, which describe how to create an Oracle7 database, should be followed in the order presented.

To Create a New Database and Make It Available for System Use

1. Back up any existing databases.
2. Create parameter files.
3. Edit new parameter files.
4. Check the instance identifier for your system.
5. Start Server Manager and connect to Oracle7 as an administrator.
6. Start an instance.
7. Create the database.
8. Back up the database.

See Also: These steps provide general information about database creation on all operating systems. See your operating system–specific Oracle documentation for information about creating databases on your platform.

Step 1 **Back up any existing databases.**

Oracle Corporation strongly recommends that you make complete backups of all existing databases before creating a new database, in case database creation accidentally affects some existing files. Backup should include parameter files, datafiles, redo log files, and control files.

Step 2 **Create parameter files.**

The instance (System Global Area and background processes) for any Oracle7 database is started using a parameter file.

Each database on your system should have at least one customized parameter file that corresponds only to that database. Do not use the same file for several databases.

To create a parameter file for the database you are about to make, use your operating system to make a copy of the parameter file that Oracle7 provided on the distribution media. Give this copy a new filename. You can then edit and customize this new file for the new database.

See Also: For more information about copying the parameter file, see your operating system–specific Oracle documentation.

> **Note:** In distributed processing environments, Server Manager is often executed from a client machine of the network. If a client machine is being used to execute Server Manager and create a new database, you need to copy the new parameter file (currently located on the computer executing Oracle7) to your client workstation. This procedure is operating system–dependent. For more information about copying files among the computers of your network, see your operating system–specific Oracle documentation.

Step 3 Edit new parameter files.

To create a new database, inspect and edit the following parameters of the new parameter file:

Parameter	Described on
DB_NAME	page 2 – 9
DB_DOMAIN	page 2 – 9
CONTROL_FILES	page 2 – 10
DB_BLOCK_SIZE	page 2 – 11
DB_BLOCK_BUFFERS	page 2 – 11
PROCESSES	page 2 – 12
ROLLBACK_SEGMENTS	page 2 – 12

Table 2 – 1 Suggested Initialization Parameters to Edit

You should also edit the appropriate license parameter(s):

Parameter	Described on
LICENSE_MAX_SESSIONS	page 2 – 12
LICENSE_SESSION_WARNING	page 2 – 12
LICENSE_MAX_USERS	page 2 – 13

Table 2 – 2 License Initialization Parameters

Step 4 Check the instance identifier for your system.

If you have other databases, check the Oracle7 instance identifier. The Oracle7 instance identifier should match the name of the database (the value of DB_NAME) to avoid confusion with other Oracle7 instances that are running concurrently on your system.

See your operating system–specific Oracle documentation for more information.

Step 5 Start Server Manager and connect to Oracle7 as an administrator.

Once Server Manager is running, connect to the database as an administrator.

See Also: Starting Server Manager is operating system specific; see your operating system–specific Oracle documentation for details.

Step 6 Start an instance.

To start an instance (System Global Area and background processes) to be used with the new database, use the Startup Database dialog box of Server Manager. In the Startup Database dialog box, make sure that you have selected the Startup Nomount radio button.

After selecting the Startup Nomount, the instance starts. At this point, there is no database. Only an SGA and background processes are started in preparation for the creation of a new database.

Step 7 Create the database.

To create the new database, use the SQL command CREATE DATABASE, optionally setting parameters within the statement to name the database, establish maximum numbers of files, name the files and set their sizes, and so on.

When you execute a CREATE DATABASE statement, Oracle performs the following operations:

- creates the datafiles for the database
- creates the control files for the database
- creates the redo log files for the database
- creates the SYSTEM tablespace and the SYSTEM rollback segment
- creates the data dictionary
- creates the users SYS and SYSTEM
- specifies the character set that stores data in the database
- mounts and opens the database for use

⚠ **Warning:** Make sure that the datafiles and redo log files that you specify do not conflict with files of another database.

Step 8 Back up the database.

You should make a full backup of the database to ensure that you have a complete set of files from which to recover if a media failure occurs. See Chapter 23.

See Also: "Backing Up a Database," Chapter 23.

"Using Parameter Files" on page 3 – 10 for more information about parameter files.

Oracle7 Server SQL Reference for information about the CREATE DATABASE command, character sets, and database creation.

Creating a Database: Example

The following statement is an example of a CREATE DATABASE statement:

```
CREATE DATABASE test
   LOGFILE
     GROUP 1 ('test_log1a', 'test_log1b') SIZE 500K,
     GROUP 2 ('test_log2a', 'test_log2b') SIZE 500K,
   DATAFILE 'test_system' SIZE 10M;
```

The values of the MAXLOGFILES, MAXLOGMEMBERS, MAXDATAFILES, MAXLOGHISTORY, and MAXINSTANCES options in this example assume the default values, which are operating system–dependent. The database is mounted in the default modes NOARCHIVELOG and EXCLUSIVE and then opened.

The items and information in the example statement above result in creating a database with the following characteristics:

- The new database is named TEST.
- The SYSTEM tablespace of the new database is comprised of one 10 MB datafile named TEST_SYSTEM.
- The new database has two online redo log groups, each containing two 500 KB members.
- The new database *does not* overwrite any existing files to create the control files specified in the parameter file.

> **Note:** You can set several limits during database creation. Some of these limits are also subject to superseding limits of the operating system and can affect each other. For example, if you set MAXDATAFILES, Oracle7 allocates enough space in the control file to store MAXDATAFILES filenames, even if the database has only one datafile initially; because the maximum control file size is limited and operating system–dependent, you might not be able to set all CREATE DATABASE parameters at their theoretical maximums.

See Also: For more information about setting limits during database creation, see the *Oracle7 Server SQL Reference.*

See your operating system–specific Oracle documentation for information about operating system limits.

Troubleshooting Database Creation

If for any reason database creation fails, shut down the instance and delete any files created by the CREATE DATABASE statement before you attempt to create it once again.

After correcting the error that caused the failure of the database creation, return to Step 6 of "Creating a Oracle7 Database."

Dropping a Database

To drop a database, remove its datafiles, redo log files, and all other associated files (control files, parameter files, archived log files).

To view the names of the database's datafiles and redo log files, query the data dictionary views V$DBFILE and V$LOGFILE.

See Also: For more information about these views, see the *Oracle7 Server Reference.*

Parameters

As described in Step 3 of “Creating an Oracle7 Database”, Oracle suggests you alter a minimum set of parameters. These parameters are described in the following sections:

- DB_NAME and DB_DOMAIN
- CONTROL_FILES
- DB_BLOCK_SIZE
- DB_BLOCK_BUFFERS
- PROCESSES
- ROLLBACK_SEGMENTS
- License Parameters
- LICENSE_MAX_SESSIONS and LICENSE_SESSIONS_WARNING
- LICENSE_MAX_USERS

DB_NAME and DB_DOMAIN

A database’s *global database name* (name and location within a network structure) is created by setting both the DB_NAME and DB_DOMAIN parameters before database creation. After creation, the database’s name cannot be easily changed. The DB_NAME parameter determines the local name component of the database’s name, while the DB_DOMAIN parameter indicates the domain (logical location) within a network structure. The combination of the settings for these two parameters should form a database name that is unique within a network. For example, to create a database with a global database name of TEST.US.ACME.COM, edit the parameters of the new parameter file as follows:

```
DB_NAME = TEST
DB_DOMAIN = US.ACME.COM
```

DB_NAME must be set to a text string of no more than eight characters. During database creation, the name provided for DB_NAME is recorded in the datafiles, redo log files, and control file of the database. If during database instance startup the value of the DB_NAME parameter (of the parameter file) and the database name in the control file are not the same, the database does not start.

DB_DOMAIN is a text string that specifies the network domain where the database is created; this is typically the name of the organization that owns the database. If the database you are about to create will ever be part of a distributed database system, pay special attention to this initialization parameter before database creation.

See Also: For more information about distributed databases, see *Oracle7 Server Distributed Systems, Volume I.*

CONTROL_FILES

Include the CONTROL_FILES parameter in your new parameter file and set its value to a list of control filenames to use for the new database. If you want Oracle7 to create new operating system files when creating your database's control files, make sure that the filenames listed in the CONTROL_FILES parameter do not match any filenames that currently exist on your system. If you want Oracle7 to reuse or overwrite existing files when creating your database's control files, make sure that the filenames listed in the CONTROL_FILES parameter match the filenames that currently exist.

> ⚠ **Warning:** Use extreme caution when setting this option. If you inadvertently specify a file that you did not intend and execute the CREATE DATABASE statement, the previous contents of that file will be overwritten.

If no filenames are listed for the CONTROL_FILES parameter, Oracle7 uses a default filename.

Oracle Corporation strongly recommends you use at least two control files stored on separate physical disk drives for each database. Therefore, when specifying the CONTROL_FILES parameter of the new parameter file, follow these guidelines:

- List at least two filenames for the CONTROL_FILES parameter.
- Place each control file on a separate physical disk drives by fully specifying filenames that refer to different disk drives for each filename.

> **Note:** The file specification for control files is operating system–dependent. Regardless of your operating system, *always* fully specify filenames for your control files.

When you execute the CREATE DATABASE statement (in Step 7), the control files listed in the CONTROL_FILES parameter of the parameter file will be created.

See Also: The default filename for the CONTROL_FILES parameter is operating system–dependent. See your operating system–specific Oracle documentation for details.

DB_BLOCK_SIZE

The default data block size for every Oracle7 Server is operating system–specific. The Oracle7 data block size is typically either 2K or 4K. Generally, the default data block size is adequate. In some cases, however, a larger data block size provides greater efficiency in disk and memory I/O (access and storage of data). Such cases include:

- Oracle7 is on a large computer system with a large amount of memory and fast disk drives. For example, databases controlled by mainframe computers with vast hardware resources typically use a data block size of 4K or greater.
- The operating system that runs Oracle7 uses a small operating system block size. For example, if the operating system block size is 1K and the data block size matches this, Oracle7 may be performing an excessive amount of disk I/O during normal operation. To correct for this, all databases created should have a data block size that is larger than the operating system block size.

Each database's block size is set during database creation by the initialization parameter DB_BLOCK_SIZE. The block size *cannot* be changed after database creation except by re–creating the database. If a database's block size is different from the operating system block size, make the data block size a multiple of the operating system's block size.

For example, if your operating system's block size is 2K (2048 bytes), the following setting for the DB_BLOCK_SIZE initialization parameter would be valid:

```
DB_BLOCK_SIZE=4096
```

DB_BLOCK_SIZE also determines the size of the database buffers in the buffer cache of the System Global Area (SGA).

See Also: For details about your default block size, see your operating system–specific Oracle documentation.

DB_BLOCK_BUFFERS

This parameter determines the number of buffers in the buffer cache in the System Global Area (SGA). The number of buffers affects the performance of the cache. Larger cache sizes reduce the number of disk writes of modified data. However, a large cache may take up too much memory and induce memory paging or swapping.

Estimate the number of data blocks that your application accesses most frequently, including tables, indexes, and rollback segments. This estimate is a rough approximation of the minimum number of buffers the cache should have. Typically, 1000 or 2000 buffers is sufficient.

See Also: For more information about tuning the buffer cache, see the *Oracle7 Server Tuning* manual.

PROCESSES

This parameter determines the maximum number of operating system processes that can be connected to Oracle7 concurrently. The value of this parameter must include 5 for the background processes and 1 for each user process. For example, if you plan to have 50 concurrent users, set this parameter to at least 55.

ROLLBACK_ SEGMENTS

This parameter is a list of the rollback segments an Oracle7 instance acquires at database startup. List your rollback segments as the value of this parameter.

> ☞ **Attention:** After installation, you must create at least one rollback segment in the SYSTEM tablespace in addition to the SYSTEM rollback segment before you can create any schema objects.

See Also: For more information about how many rollback segments you need, see *Oracle7 Server Tuning*.

License Parameters

Oracle7 helps you ensure that your site complies with its Oracle7 license agreement. If your site is licensed by concurrent usage, you can track and limit the number of sessions concurrently connected to an instance. If your site is licensed by named users, you can limit the number of named users created in a database. To use this facility, you need to know which type of licensing agreement your site has and what the maximum number of sessions or named users is. Your site might use either type of licensing (session licensing or named user licensing), but not both.

See Also: For more information about managing licensing, see page 19 – 2.

LICENSE_MAX_ SESSIONS and LICENSE_SESSIONS_ WARNING

You can set a limit on the number of concurrent sessions that can connect to a database on the specified computer. To set the maximum number of concurrent sessions for an instance, set the parameter LICENSE_MAX_SESSIONS in the parameter file that starts the instance, as shown in the following example:

```
LICENSE_MAX_SESSIONS = 80
```

In addition to setting a maximum number of sessions, you can set a warning limit on the number of concurrent sessions. Once this limit is reached, additional users can continue to connect (up to the maximum limit), but Oracle7 sends a warning for each connecting user. To set the warning limit for an instance, set the parameter LICENSE_SESSIONS_WARNING. Set the warning limit to a value lower than LICENSE_MAX_SESSIONS.

For instances running with the Parallel Server, each instance can have its own concurrent usage limit and warning limit. However, the sum of the instances' limits must not exceed the site's session license.

See Also: For more information about setting these limits when using the Parallel Server, see *Oracle7 Parallel Server Concepts & Administration*.

LICENSE_MAX_USERS

You can set a limit on the number of users created in the database. Once this limit is reached, you cannot create more users.

> **Note:** This mechanism assumes that each person accessing the database has a unique user name and that no people share a user name. Therefore, so that named user licensing can help you ensure compliance with your Oracle7 license agreement, do not allow multiple users to log in using the same user name.

To limit the number of users created in a database, set the LICENSE_MAX_USERS parameter in the database's parameter file, as shown in the following example:

```
LICENSE_MAX_USERS = 200
```

For instances running with the Parallel Server, all instances connected to the same database should have the same named user limit.

See Also: For more information about setting this limit when using the Parallel Server see the *Oracle7 Parallel Server Concepts & Administration* manual.

Considerations After Creating a Database

After you create a database, the instance is left running, and the database is open and available for normal database use. Use Server Manager to subsequently start and stop the database. If more than one database exists in your database system, specify the parameter file to use with any subsequent database startup.

If you plan to install other Oracle products to work with this database, see the installation instructions for those products; some products require you to create additional data dictionary tables. See your operating system–specific Oracle documentation for the additional products. Usually, command files are provided to create and load these tables into the database's data dictionary.

The Oracle7 Server distribution media can include various SQL files that let you experiment with the system, learn SQL, or create additional tables, views, or synonyms.

A newly created database has only two users, SYS and SYSTEM. The passwords for these two usernames should be changed soon after the database is created.

See Also: For more information about the users SYS and SYSTEM see "Database Administrator Usernames" on page 1 – 4.

For information about changing a user's password see "Altering Users" on page 19 – 12.

Initial Tuning Guidelines

You can make a few significant tuning alterations to Oracle7 immediately following installation. By following these instructions, you can reduce the need to tune Oracle7 when it is running. This section gives recommendations for the following installation issues:

- Allocating Rollback Segments
- Choosing Sizes for Rollback Segments
- Choosing the Number of DB_BLOCK_LRU_ LATCHES
- Distributing I/O

Allocating Rollback Segments

Proper allocation of rollback segments makes for optimal database performance. The size and number of rollback segments required for optimal performance depends on your application. The *Oracle7 Server Tuning* manual contains some general guidelines for choosing how many rollback segments to allocate based on the number of concurrent transactions on your Oracle7 Server. These guidelines are appropriate for most application mixes.

To create rollback segments, use the CREATE ROLLBACK SEGMENT command.

See Also: For information about the CREATE ROLLBACK SEGMENT command, see the *Oracle7 Server SQL Reference*.

Choosing Sizes for Rollback Segments

The size of your rollback segment can also affect performance. Rollback segment size is determined by the storage parameters in the CREATE ROLLBACK SEGMENT statement. Your rollback segments must be large enough to hold the rollback entries for your transactions.

See Also: For information about choosing sizes for your rollback segments, see the *Oracle7 Server Tuning* manual.

Choosing the Number of DB_BLOCK_LRU_ LATCHES

Contention for the LRU latch can impede performance on symmetric multiprocessor (SMP) machines with a large number of CPUs. The LRU latch controls the replacement of buffers in the buffer cache. For SMP systems, Oracle automatically sets the number of LRU latches to be one half the number of CPUs on the system. For non–SMP systems, one LRU latch is sufficient.

You can specify the number of LRU latches on your system with the initialization parameter DB_BLOCK_LRU_LATCHES. This parameter sets the maximum value for the desired number of LRU latches. Each LRU latch will control a set of buffers and Oracle balances allocation of replacement buffers among the sets.

See Also: For more information on LRU latches, see the *Oracle7 Server Tuning* manual.

Distributing I/O

Proper distribution of I/O can improve database performance dramatically. I/O can be distributed during installation of Oracle7. Distributing I/O during installation can reduce the need to distribute I/O later when Oracle7 is running.

There are several ways to distribute I/O when you install Oracle7:

- redo log file placement
- datafile placement
- separation of tables and indexes
- density of data (rows per data block)

See Also: For information about ways to distribute I/O, see the *Oracle7 Server Tuning* manual.

CHAPTER

3

Starting Up and Shutting Down

This chapter describes the procedures for starting and stopping an Oracle7 database, and includes the following topics:

- Startup Procedures
- Altering Database Availability
- Shutdown Procedures
- Using Parameter Files

See Also: *Trusted Oracle7 Server Administrator's Guide*, for more information about starting up and shutting down Trusted Oracle7.

Oracle Server Manager User's Guide, for more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode.

Startup Procedures

This section includes the following topics:

- Preparing to Start an Instance
- Starting an Instance: Scenarios

To start up a database or instance, use either the Server Manager Startup Database dialog box or the STARTUP command (after you connect to Oracle7 with administrator privileges). You can start an instance and database in a variety of ways:

- start the instance without mounting a database
- start the instance and mount the database, but leave it closed
- start the instance, and mount and open the database in:
  - unrestricted mode (accessible to all users)
  - RESTRICTED mode (accessible to DBAs only)

☞ **Attention:** You cannot start a database instance if you are connected to the database via a multi–threaded server process.

In addition, you can force the instance to start, or start the instance and have complete media recovery begin immediately. If your operating system supports the Oracle7 Parallel Server, you may start an instance and mount the database in either exclusive or shared mode.

See Also: *Trusted Oracle7 Server Administrator's Guide*, for more information about database startup and Trusted Oracle7.

Preparing to Start an Instance

There are several tasks you need to perform before you attempt to start an instance.

To Prepare to Start an Instance

1. Start Server Manager and connect with administrator privileges.

   To start up a database or instance, you must start Server Manager. You must also be connected with administrator privileges.

2. Specify a database name.

   When starting a database instance, specify the name of the database that will be mounted to the instance by either:

   - using the STARTUP command and specifying the database name

- specifying DB_NAME in the parameter file that starts the instance

3. Specify the parameter filename.

   When starting a database instance, choose a parameter file to initialize the instance's settings:

   - using the Startup Database dialog box and entering a filename in the Parameter File text entry field
   - using the STARTUP command with the PFILE option and a fully specified filename

See Also: The specification of filenames is operating system–specific. See your operating system–specific Oracle documentation. If no filename is entered, Oracle7 uses the default filename.

Starting an Instance: Scenarios

The following scenarios describe the many ways in which you can start up an instance.

> **Note:** You may encounter problems starting up an instance if control files, database files, or redo log files are not available. If one or more of the files specified by the CONTROL_FILES parameter do not exist or cannot be opened when you attempt to mount a database, Oracle7 returns a warning message and does not mount the database. If one or more of the datafiles or redo log files is not available or cannot be opened when attempting to open a database, Oracle7 returns a warning message and does not open the database.

Starting an Instance Without Mounting a Database

You might want to start an instance without mounting a database; this is usually the case only during database creation. To do this, use one of the following options of Server Manager:

- the Startup Database dialog box, selecting the Startup Nomount radio button
- the STARTUP command with the NOMOUNT option

Starting an Instance and Mounting a Database

You might want to start an instance and mount a database, but not open the database because you want to perform specific maintenance operations. For example, the database must be mounted but not open during the following tasks:

- renaming datafiles
- adding, dropping, or renaming redo log files

- enabling and disabling redo log archiving options
- performing full database recovery

Start an instance and mount the database, but leave it closed using one of the following options of Server Manager:

- the Startup database dialog box, selecting the Startup Mount radio button
- the STARTUP command with the MOUNT option

Starting an Instance, and Mounting and Opening a Database

Normal database operation means that an instance is started and the database is mounted and open; this allows any valid user to connect to the database and perform typical data access operations.

Start an instance, and mount and open the database, using one of the following options of Server Manager:

- the Startup Database dialog box, selecting the Startup Open radio button
- the STARTUP command with the OPEN option

Restricting Access to a Database at Startup

You might want to start an instance, and mount and open a database in restricted mode so that the database is available only to administrative personnel (not general database users). Use this mode of database startup when you need to accomplish one of the following tasks:

- perform structure maintenance, such as rebuilding indexes
- perform an export or import of database data
- perform a data load (with SQL*Loader)
- temporarily prevent typical users from using data

Typically, all users with the CREATE SESSION system privilege can connect to an open database. Opening a database in restricted mode allows database access only to users with both the CREATE SESSION and RESTRICTED SESSION system privilege; only database administrators should have the RESTRICTED SESSION system privilege.

Start an instance (and, optionally, mount and open the database) in restricted mode using one of the following options of Server Manager:

- the Startup Database dialog box, selecting the Restrict button
- the STARTUP command with the RESTRICT option

Later, you can make the database accessible to users who do not have the RESTRICTED SESSION system privilege.

Forcing an Instance to Start

In unusual circumstances, you might experience problems when attempting to start a database instance. A database instance should not be forced to start unless you are faced with the following:

- The current instance cannot be successfully shut down using either the Normal or Immediate radio buttons of the Shutdown Database dialog box (or an equivalent SHUTDOWN statement).
- You experience problems when starting an instance.

If one of these situations arises, you can usually solve the problem by starting a new instance (and optionally mounting and opening the database) using either of the following options of Server Manager:

- the Startup Database dialog box with the Force button selected
- the STARTUP command with the FORCE option

Starting an Instance, Mounting a Database, and Starting Complete Media Recovery

If you know that media recovery is required, you can start an instance, mount a database to the instance, and have the recovery process automatically start by using the STARTUP command with the RECOVER option.

Starting in Exclusive or Parallel Mode

If your Oracle7 Server allows multiple instances to access a single database concurrently, you must choose whether to mount the database exclusively or in parallel.

Starting Up an Instance and Database: Example

The following statement starts an instance using the parameter file INITSALE.ORA, mounts and opens the database named SALES in exclusive mode, and restricts access to administrative personnel. The DBA is already connected with administrator privileges.

```
STARTUP OPEN sales PFILE=INITSALE.ORA EXCLUSIVE RESTRICT;
```

Automatic Database Startup at Operating System Start

Many sites use procedures to enable automatic startup of one or more Oracle7 instances and databases immediately following a system start. The procedures for doing this are specific to each operating system.

Starting Remote Instances

If your local Oracle7 Server is part of a distributed database, you might need to start a remote instance and database. Procedures for starting and stopping remote instances vary widely depending on communication protocol and operating system.

See Also: For more information about making a database available to non–privileged users, see “Restricting Access to an Open Database” on page 3 – 7.

For more information about recovering control files, database files and redo logs, see Chapter 24.

For more information about the side effects of aborting the current instance, see “Aborting an Instance” on page 3 – 9.

For more information about starting up in exclusive or parallel mode, see the *Oracle7 Parallel Server Concepts & Administration* manual.

For more information about the restrictions that apply when combining options of the STARTUP command, see the *Oracle7 Server SQL Reference.*

For more information about automatic startup procedure topics, see your operating system–specific Oracle documentation.

Altering Database Availability

You can make a database partially available by opening a previously mounted but closed database so that users can connect to and use the database.

The following sections explain how to alter a database’s availability:

- Mounting a Database to an Instance
- Opening a Closed Database
- Restricting Access to an Open Database

Mounting a Database to an Instance

When you need to perform specific administrative operations, the database must be started and mounted to an instance, but closed. This can be accomplished by starting the instance and mounting the database.

When mounting the database, you can indicate whether to mount the database exclusively to this instance or concurrently to other instances.

To mount a database to a previously started instance, use either of the following options:

- the Mount menu item of Server Manager
- the SQL command ALTER DATABASE with the MOUNT option

Use the following statement when you want to mount a database in exclusive mode:

```
ALTER DATABASE MOUNT;
```

See Also: For a list of operations that require the database to be mounted and closed, (and procedures to start an instance and mount a database in one step) see “Starting an Instance and Mounting a Database” on page 3 – 3.

Opening a Closed Database

You can make a mounted but closed database available for general use by opening the database. To open a mounted database, use either of the following options:

- the Open menu item of Server Manager
- the SQL command ALTER DATABASE with the OPEN option

Use the following statement to open a mounted database:

```
ALTER DATABASE OPEN;
```

After executing this statement, any valid Oracle7 user with the CREATE SESSION system privilege can connect to the database.

Restricting Access to an Open Database

Under normal conditions, all users with the CREATE SESSION system privilege can connect to an instance. However, you can take an instance in and out of restricted mode. When an instance is in restricted mode, only users who have both the CREATE SESSION and RESTRICTED SESSION system privileges can connect to it. Typically, only administrators have the RESTRICTED SESSION system privilege.

Restricted mode is useful when you need to perform the following tasks:

- perform structure maintenance, such as rebuilding indexes
- perform an export or import of database data
- perform a data load (with SQL*Loader)
- temporarily prevent non–administrator users from using data

To place an instance in restricted mode, use the Restrict menu item of Server Manager or the SQL command ALTER SYSTEM with the ENABLE RESTRICTED SESSION option. After placing an instance in restricted mode, you might want to kill all current user sessions before performing any administrative tasks.

To lift an instance from restricted mode, use the Allow All menu item of Server Manager or the SQL command ALTER SYSTEM with the DISABLE RESTRICTED SESSION option.

See Also: For more information about killing sessions, see “Terminating Sessions” on page 4 – 16.

For more information about starting a database instance, and mounting and opening the database in restricted mode, see “Restricting Access to a Database at Startup” on page 3 – 4.

Shutdown Procedures

The following sections describe various shutdown procedures:

- Shutting Down a Database Under Normal Conditions
- Shutting Down a Database Immediately
- Aborting an Instance

To initiate database shutdown, use either the Shutdown Database dialog box of Server Manager or the SQL command SHUTDOWN. Control is not returned to the session that initiates a database shutdown until shutdown is complete. Users who attempt connections while a shutdown is in progress receive a message like the following:

```
ORA–01090: shutdown in progress – connection is not permitted
```

☞ **Attention:** You cannot shut down a database if you are connected to the database via a multi–threaded server process.

To shut down a database and instance, you must first be connected with administrator privileges. This condition applies whether you are using Server Manager / GUI or SQL commands.

See Also: Several special options and conditions of database shutdown that apply when using Trusted Oracle7 in OS MAC mode are not discussed in this section. For more information about database shutdown and Trusted Oracle7, see the *Trusted Oracle7 Server Administrator's Guide.*

Shutting Down a Database Under Normal Conditions

Normal database shutdown proceeds with the following conditions:

- No new connections are allowed after the statement is issued.
- Before the database is shut down, Oracle7 waits for all currently connected users to disconnect from the database.
- The next startup of the database will not require any instance recovery procedures.

To shut down a database in normal situations, use either of the following options of Server Manager:

- the Normal radio button of the Shutdown Database dialog box
- the SHUTDOWN command with the NORMAL option (`SHUTDOWN NORMAL;`)

Shutting Down a Database Immediately

Use immediate database shutdown only in the following situations:

- A power shutdown is going to occur soon.
- The database or one of its applications is functioning irregularly.

Immediate database shutdown proceeds with the following conditions:

- Current client SQL statements being processed by Oracle7 are terminated immediately.
- Any uncommitted transactions are rolled back. (If long uncommitted transactions exist, this method of shutdown might not complete quickly, despite its name.)
- Oracle7 does not wait for users currently connected to the database to disconnect; Oracle7 implicitly rolls back active transactions and disconnects all connected users.

To shut down a database immediately, use either of the following options of Server Manager:

- the Immediate radio button of the Shutdown database dialog box
- the SHUTDOWN command with the IMMEDIATE option

Aborting an Instance

You can shutdown a database instantaneously by aborting the database's instance. If possible, perform this type of shutdown *only* when in the following situations:

- The database or one of its applications is functioning irregularly *and* neither of the other types of shutdown work.
- You need to shut down the database instantaneously (for example, if you know a power shutdown is going to occur in one minute).
- You experience problems when starting a database instance.

Aborting an instance shuts down a database and yields the following results:

- Current client SQL statements being processed by Oracle7 are immediately terminated.
- Uncommitted transactions are not rolled back.
- Oracle7 does not wait for users currently connected to the database to disconnect; Oracle7 implicitly disconnects all connected users.

If *both* the normal and immediate shutdown options do not work, abort the current database instance immediately by using either of the following options of Server Manager:

- the Abort radio button of the Shutdown Database dialog box
- the SHUTDOWN command with the ABORT option

Using Parameter Files

The following sections include information about how to use parameter files:

- The Sample Parameter File
- The Number of Parameter Files
- The Location of the Parameter File in Distributed Environments

To start an instance, Oracle7 must read a *parameter file,* which is a text file containing a list of instance configuration parameters. Often, although not always, this file is named INIT.ORA or INIT*sid*.ORA, where *sid* is operating system–specific.

You can edit parameter values in a parameter file with a basic text editor; however, editing methods are operating system–specific.

Oracle7 treats string literals defined for National Language Support (NLS) parameters in the file as if they are in the database character set.

See Also: For more information about INIT*sid*.ORA, see your operating system–specific Oracle documentation.

The Sample Parameter File

A sample parameter file (INIT.ORA or INIT*sid*.ORA) is included in the Oracle7 distribution set. This sample file's parameters are adequate for initial installations of an Oracle7 database. After your system is operating and you have some experience with Oracle7, you will probably want to change some parameter values.

See Also: For more information about optimizing a database's performance using the parameter file, see the *Oracle7 Server Tuning* manual.

The Number of Parameter Files

Each Oracle7 database has at least one parameter file that corresponds only to that database. This way, database–specific parameters (such as DB_NAME and CONTROL_FILES) in a given file always pertain to a particular database. It is also possible to have several different parameter files for a single database. For example, you can have several different parameter files for a single database so you can optimize the database's performance in different situations.

The Location of the Parameter File in Distributed Environments

Server Manager must be able to read a database's parameter file to start a database's instance. Therefore, always store a database's parameter file on the computer executing Server Manager.

For example, in non–distributed processing installations, the same computer executes Oracle7 and Server Manager; therefore, this computer has the parameter file stored on one of its disk drives.

However, in distributed processing installations, local client workstations can execute Server Manager to administer a database stored on a remote machine. In this type of configuration, the local client machines must each store a copy of the parameter file for the corresponding databases.

See Also: For more information about using administering Oracle7 in a distributed environment, see *Oracle7 Server Distributed Systems, Volume I.*

For information concerning the setup and implementation of Server Manager, see your operating system–specific Oracle documentation.

PART

II

Oracle Server Configuration

CHAPTER

4

Managing Oracle7 Processes

This chapter describes how to manage the processes of an Oracle7 instance, and includes the following topics:

- Configuring Oracle7 for Dedicated Server Processes
- Configuring Oracle7 for Multi–Threaded Server Processes
- Modifying Server Processes
- Tracking Oracle7 Processes
- Managing Processes for the Parallel Query Option
- Terminating Sessions

See Also: For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide.*

Configuring Oracle7 for Dedicated Server Processes

When a user process executes the database application on one machine, and a server process executes the associated Oracle server on another machine, you have separate, distinct processes. The separate server process created on behalf of each user is a *dedicated server process* (see Figure 4 – 1). Oracle7 is automatically installed for this configuration. If your operating system can support Oracle7 in this configuration, it may also support multi–threaded server processes.

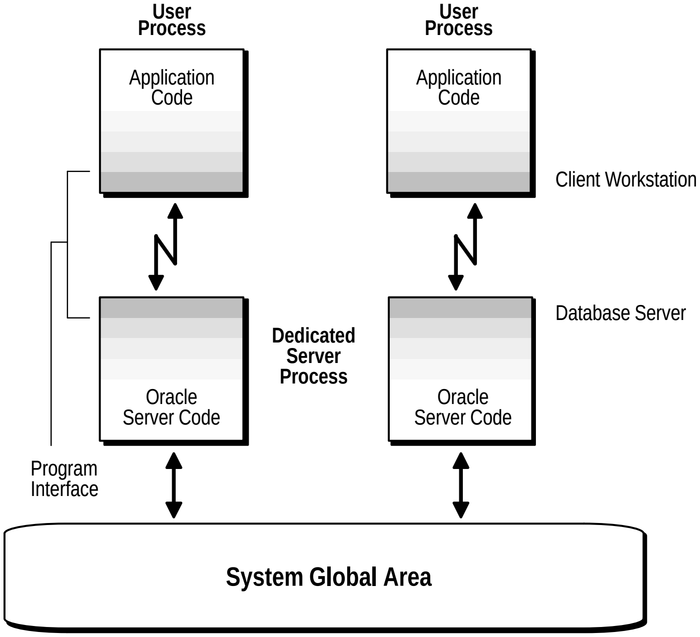


Figure 4 – 1 Oracle Dedicated Server Processes

To start an instance in a dedicated server configuration, set the following initialization parameters (in the parameter file) to ”null”, or omit them from the file altogether:

- MTS_SERVICE
- MTS_DISPATCHERS
- MTS_SERVERS
- MTS_LISTENER_ADDRESS

When to Connect to a Dedicated Server Process

If possible, users should connect to an instance via a dispatcher. This keeps the number of processes required for the running instance low. In the following situations, however, users and administrators should explicitly connect to an instance using a dedicated server process:

- to submit a batch job (for example, when a job can allow little or no idle time for the server process)
- to use Server Manager to start up, shut down, or perform media recovery on a database

To request a dedicated server connection, users must include the SRVR=DEDICATED clause in their SQL*Net TNS connect string.

See Also: For a complete description of SQL*Net connect string syntax, see your operating system–specific Oracle documentation and your SQL*Net documentation.

For more information about initialization parameters and parameter files, see the *Oracle7 Server Reference.*

Configuring Oracle7 for Multi–Threaded Server Processes

Consider an order entry system with dedicated server processes. A customer places an order as a clerk enters the order into the database. For most of the transaction, the clerk is on the telephone talking to the customer and the server process dedicated to the clerk's user process remains idle. The server process is not needed during most of the transaction, and the system is slower for other clerks entering orders.

The *multi–threaded server* configuration eliminates the need for a dedicated server process for each connection (see Figure 4 – 2). A small number of shared server processes can perform the same amount of processing as many dedicated server processes. Also, the amount of memory required for each user is relatively small. Because less memory and process management are required, more users can be supported.

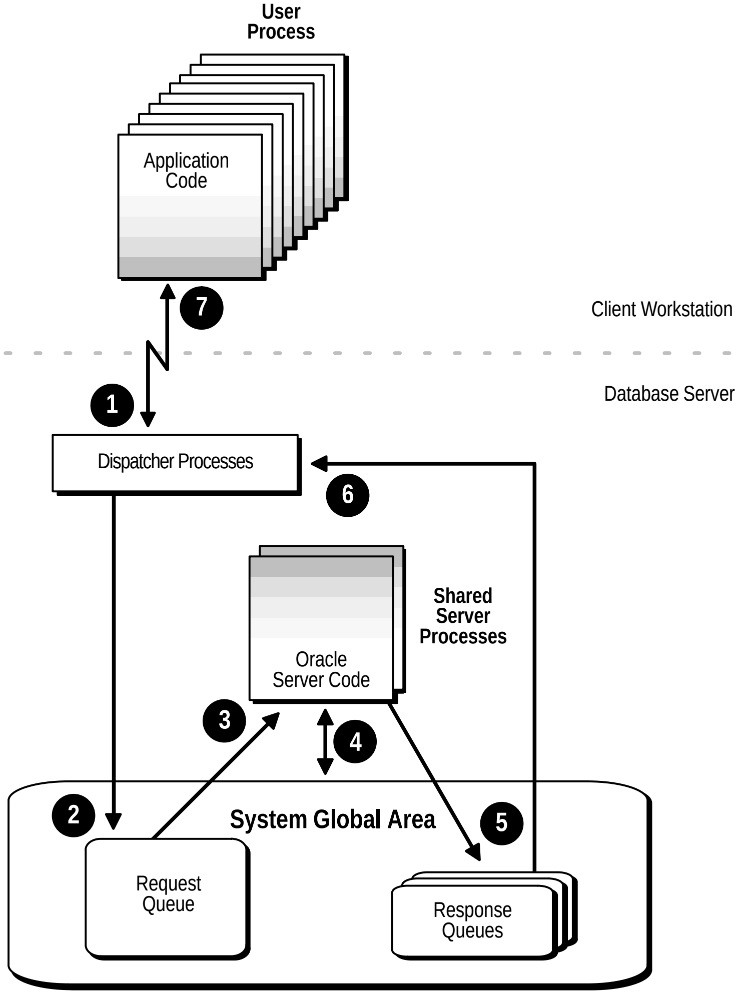


Figure 4 – 2 Oracle Multi–Threaded Sever Processes

To set up your system in a multi–threaded server configuration, start a network listener process and set the following initialization parameters:

- SHARED_POOL_SIZE
- MTS_LISTENER_ADDRESS
- MTS_SERVICE
- MTS_DISPATCHERS

- MTS_MAX_DISPATCHERS
- MTS_SERVERS
- MTS_MAX_SERVERS

After setting these initialization parameters, restart the instance, which at this point will use the multi–threaded server configuration. The multi–threaded server architecture requires SQL*Net Version 2. User processes targeting the multi–threaded server must connect through SQL*Net, even if they are on the same machine as the Oracle7 instance.

See Also: For more information about starting and managing the network listener process, see *Oracle7 Server Distributed Systems, Volume I* and the *Oracle Network Manager Administrator's Guide.*

SHARED_POOL_SIZE: Allocating Additional Space in the Shared Pool for Shared Server

When users connect through the multi–threaded server, Oracle7 needs to allocate additional space in the shared pool for storing information about the connections between the user processes, dispatchers, and servers. For each user who will connect using the multi–threaded server, add 1K to the setting of the parameter SHARED_POOL_SIZE.

See Also: For more information about this parameter, see the *Oracle7 Server Reference.*

For more information about tuning, see the *Oracle7 Server Tuning* manual.

MTS_LISTENER_ADDRESS: Setting the Listener Process Address

Within the database's parameter file, set the initialization parameter MTS_LISTENER_ADDRESS for each port to which the database will connect. The parameter supports the following syntax:

```
MTS_LISTENER_ADDRESS = ”(addr)”
```

In the syntax above, *addr* is an address at which the listener will listen for connection requests for a specific protocol. The parameter file may contain multiple addresses.

The following examples specify listener addresses:

```
MTS_LISTENER_ADDRESS = ”(ADDRESS=(PROTOCOL=tcp)(PORT=5000)\
   (HOST=ZEUS)”
MTS_LISTENER_ADDRESS = ”(ADDRESS=(PROTOCOL=decnet)\
   (OBJECT=OUTA)(NODE=ZEUS)”
```

Each address specified in the database's parameter file must also be specified in the corresponding listener's configuration file. You specify addresses differently for various network protocols.

See Also: For more information about specifying addresses for the network listener process, see your operating system–specific Oracle documentation and your SQL*Net documentation.

MTS_SERVICE: Specifying Service Names for Dispatchers

Specify the name of the service associated with dispatchers using the parameter MTS_SERVICE. A user requests the multi–threaded server by specifying this service name in the connect string. A service name must be unique; if possible, use the instance's SID (system identifier).

If you do not set the MTS_SERVICE parameter, its value defaults to the DB_NAME parameter. (If DB_NAME is also not set, Oracle7 returns the error ORA–00114, "missing value for system parameter mts_service," when you start the database.)

If the dispatcher's service name is TEST_DB, the parameter would be set as follows:

```
MTS_SERVICE = "test_db"
```

A connect string for connecting to this dispatcher looks like the following:

```
SQLPLUS scott/tiger@\
   (DESCRIPTION=(ADDRESS=(PROTOCOL=decnet)(NODE=hq)\
   (OBJECT=mts7))(CONNECT_DATA=(SID=test_db)))
```

See Also: For more information about connect strings used with the multi–threaded server configuration, see your operating system–specific Oracle or SQL*Net documentation.

MTS_DISPATCHERS: Setting the Initial Number of Dispatchers

The number of dispatcher processes started at instance startup is controlled by the parameter MTS_DISPATCHERS. Estimate the number of dispatchers to start for each network protocol before instance startup.

When setting the MTS_DISPATCHERS parameter, you can include any valid protocol.

The appropriate number of dispatcher processes for each instance depends upon the performance you want from your database, the host operating system's limit on the number of connections per process, (which is operating system–dependent) and the number of connections required per network protocol.

The instance must be able to provide as many connections as there are concurrent users on the database system; the more dispatchers you have, the better potential database performance users will see, since they will not have to wait as long for dispatcher service.

After instance startup, you can start more dispatcher processes if needed; however, you can only start dispatchers that use protocols mentioned in the database's parameter file. For example, if the parameter file starts dispatchers for protocol_A and protocol_B, you cannot later start dispatchers for protocol_C without changing the parameter file and restarting the instance.

See Also: For more information about dispatcher processes, see “Adding and Removing Dispatcher Processes” on page 4 – 8.

Calculating the Initial Number of Dispatcher Processes

Once you know the number of possible connections per process for your operating system, calculate the initial number of dispatcher processes to create during instance startup, per network protocol, using the following formula Here, *connections per dispatcher* is operating system–dependent:

```
number                     maximum number of concurrent sessions
of            = CEIL   (—————————————————————————————————————————)
dispatchers                    connections per dispatcher
```

For example, assume that your system typically has 80 users concurrently connected via TCP / IP and 40 users connected via DECNet. In this case, the MTS_DISPATCHERS parameter should be set as follows:

```
MTS_DISPATCHERS = ”TCP, 3”
MTS_DISPATCHERS = ”DECNET, 3”
```

MTS_MAX_DISPATCHERS: Setting the Maximum Number of Dispatchers

The parameter MTS_MAX_DISPATCHERS sets the maximum number of dispatcher processes (of all network protocols combined) that can be started for the duration of an instance.

You can create as many dispatcher processes as you need, but the total number of processes, including dispatchers, cannot exceed the host operating system’s limit on the number of running processes.

Estimating the Maximum Number of Dispatches

To estimate the maximum number of dispatcher processes an instance will require, use the following formula:

```
                          maximum number of concurrent sessions
MTS_MAX_DISPATCHERS =     _____________________________________

                          connections per dispatcher
```

MTS_SERVERS: Setting the Initial Number of Shared Server Processes

A number of shared server processes start at instance startup, as determined by the parameter MTS_SERVERS. The appropriate number of initial shared server processes for a database system depends on how many users typically connect to it, and how much processing each user requires. If each user makes relatively few requests over a period of time, then each associated user process is idle for a large percentage of time. In that case, one shared server process can serve 10 to 20 users. If each user requires a significant amount of processing, a higher ratio of server processes to user processes is needed to handle requests.

If you want Oracle7 to use shared servers, you must set MTS_SERVERS to at least 1. If you omit the parameter or set it to 0, Oracle7 does not start any shared servers at all. However, you can subsequently set MTS_SERVERS to a number greater than 0 while the instance is running.

It is best to estimate fewer initial shared server processes. Additional shared servers start automatically when needed and are deallocated automatically if they remain idle for too long. However, the initial servers always remain allocated, even if they are idle. If you set the initial number of servers high, your system might incur unnecessary overhead. Experiment with the number of initial shared server processes and monitor shared servers until you find the ideal system performance for typical database activity.

See Also: For more information about changing the number of shared servers, see "Changing the Minimum Number of Shared Server Processes" on page 4 – 8.

MTS_MAX_SERVERS: Setting the Maximum Number of Shared Server Processes

The maximum number of shared server processes that can be started for the duration of an instance is established during instance startup by the parameter MTS_MAX_SERVERS. In general, set this parameter to allow an appropriate number of shared server processes at times of highest activity. Experiment with this limit and monitor shared servers to determine an ideal setting for this parameter.

Modifying Server Processes

This section describes changes you can make after starting an instance, and includes the following topics:

- Changing the Minimum Number of Shared Server Processes
- Adding and Removing Dispatcher Processes

Changing the Minimum Number of Shared Server Processes.

After starting an instance, you can change the minimum number of shared server processes by using the SQL command ALTER SYSTEM.

Oracle7 eventually terminates dispatchers and servers that are idle longer than the minimum limit you specify.

If you set MTS_SERVERS to 0, Oracle7 will terminate all current servers when they become idle and will not start any new servers until you increase MTS_SERVERS. Thus, setting MTS_SERVERS to 0 effectively disables the multi–threaded server temporarily.

To control the minimum number of shared server processes, you must have the ALTER SYSTEM privilege.

The following statement sets the number of shared server processes to two:

```
ALTER SYSTEM SET MTS_SERVERS = 2
```

Adding and Removing Dispatcher Processes

You can control the number of dispatcher processes in the instance. If the V$QUEUE and V$DISPATCHER views indicate that the load on the dispatcher processes is consistently high, start additional dispatcher processes to route user requests without waiting; you may start new dispatchers until the number of dispatchers equals MTS_MAX_DISPATCHER. In contrast, if the load on dispatchers is consistently low, reduce the number of dispatchers.

To change the number of dispatcher processes, use the SQL command ALTER SYSTEM. Changing the number of dispatchers for a specific protocol has no effect on dispatchers for other protocols.

You can start new dispatcher processes for protocols specified in the MTS_LISTENER_ADDRESS parameter and in the MTS_DISPATCHERS parameter. Therefore, you can add dispatchers only for protocols for which there are dispatchers; to start dispatchers for protocols for which there are currently no dispatchers, shutdown the database, change the parameter file, and restart the database.

If you reduce the number of dispatchers for a particular protocol, the dispatchers are not immediately removed. Rather, Oracle7 eventually terminates dispatchers that are idle for too long, down to the limit you specify in MTS_DISPATCHERS.

To control the number of dispatcher processes, you must have the ALTER SYSTEM privilege.

The following example adds a dispatcher process where the number of dispatchers was previously three:

```
ALTER SYSTEM
    SET MTS_DISPATCHERS = 'TCPIP,4';
```

See Also: For more information about tuning the multi–threaded server, see the *Oracle7 Server Tuning* manual.

Tracking Oracle7 Processes

An Oracle7 instance can have many background processes, which you should track if possible. This section describes how to track these processes, and includes the following topics:

- Monitoring the Processes of an Oracle7 Instance
- Trace Files, the ALERT File, and Background Processes
- Starting the Checkpoint Process

See Also: For more information about tuning Oracle7 processes, see the *Oracle7 Server Tuning* manual.

Monitoring the Processes of an Oracle7 Instance

Monitors provide a means of tracking database activity and resource usage. Selecting the Monitor feature of Server Manager / GUI displays current information about the processes of your Oracle7 database. You can operate several monitors simultaneously. Table 4 – 1 lists the Server Manager monitors that can help you track Oracle7 processes:

Monitor Name	Description
Process	The Process monitor summarizes information about all Oracle7 processes, including client–server, user, server, and background processes, currently accessing the database via the current database instance.
Session	The Session monitor shows the session ID and status of each connected Oracle7 session.

Table 4 – 1 Server Manager Monitors

Monitoring Locks

Table 4 – 2 describes two methods of monitoring locking information for ongoing transactions within an instance:

Monitor Name	Description
Server Manager Monitors	The Monitor feature of Server Manager/GUI provides two monitors for displaying lock information for an instance: Lock and Latch Monitors.
UTLLOCKT.SQL	The UTLLOCKT.SQL script displays a simple character lock wait–for graph in tree–structured fashion. Using an ad hoc query tool (such as Server Manager or SQL*Plus), the script prints the sessions in the system that is waiting for locks and the corresponding blocking locks. The location of this script file is operating system–dependent; see your operating system–specific Oracle documentation. (A second script, CATBLOCK.SQL, creates the lock views that UTLLOCKT.SQL needs, so you must run it before running UTLLOCKT.SQL.)

Table 4 – 2 Oracle7 Monitoring Facilities

Monitoring Dynamic Performance Tables

The following views, created on the dynamic performance tables, are useful for monitoring Oracle7 instance processes.

View (Monitor) Name	Description
V$CIRCUIT	Contains information about virtual circuits, which are user connections through dispatchers and servers.
V$QUEUE	Contains information about the multi–threaded message queues.
V$DISPATCHER	Contains information about dispatcher processes.
V$SHARED_SERVER	Contains information about shared server processes.
V$SQLAREA	Contains statistics about shared SQL area and contains one row per SQL string. Also provides statistics about SQL statements that are in memory, parsed, and ready for execution.
V$SESS_IO	Contains I/O statistics for each user session.
V$LATCH	Contains statistics for non–parent latches and summary statistics for parent latches.
V$SYSSTAT	Contains system statistics.

Table 4 – 3 Views for Monitoring Oracle7 Instance Processes

Following is a typical query of one of the dynamic performance tables, V$DISPATCHER. The output displays the processing load on each dispatcher process in the system:

```
SELECT (busy/(busy + idle)) * 100 ”% OF TIME BUSY”
   FROM v$dispatcher;
```

Distinguishing Oracle7 Background Processes from Operating System Background Processes

When you run many Oracle7 databases concurrently on one computer, Oracle7 provides a mechanism for naming the processes of an instance. The background process names are prefixed by an instance identifier to distinguish the set of processes for each instance.

For example, an instance named TEST might have background processes with the following names:

- ORA_TEST_DBWR
- ORA_TEST_LGWR
- ORA_TEST_SMON
- ORA_TEST_PMON
- ORA_TEST_RECO
- ORA_TEST_LCK0
- ORA_TEST_ARCH
- ORA_TEST_D000
- ORA_TEST_S000
- ORA_TEST_S001

See Also: For more information about views and dynamic performance tables see the *Oracle7 Server Reference.*

For more information about the instance identifier and the format of the Oracle7 process names, see your operating system–specific Oracle documentation.

Trace Files, the ALERT File, and Background Processes

Each server and background process can write to an associated *trace file.* When an internal error is detected by a process, it dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, while other information is for Oracle WorldWide Support. Trace file information is also used to tune applications and instances.

The *ALERT* file is a special trace file. The ALERT file of a database is a chronological log of messages and errors, which includes the following:

- all internal errors (ORA–600), block corruption errors (ORA–1578), and deadlock errors (ORA–60) that occur
- administrative operations, such as CREATE / ALTER / DROP DATABASE / TABLESPACE / ROLLBACK SEGMENT SQL statements and STARTUP, SHUTDOWN, ARCHIVE LOG, and RECOVER Server Manager statements
- several messages and errors relating to the functions of shared server and dispatcher processes
- errors occurring during the automatic refresh of a snapshot
- the values of all initialization parameters at the time the database and instance start

Oracle7 uses the ALERT file to keep a log of these special operations as an alternative to displaying such information on an operator's console (although many systems display information on the console). If an operation is successful, a "completed" message is written in the ALERT file, along with a timestamp.

Using the Trace Files

You can periodically check the ALERT file and other trace files of an instance to see if the background processes have encountered errors. For example, when the Log Writer process (LGWR) cannot write to a member of a group, an error message indicating the nature of the problem is written to the LGWR trace file and the database's ALERT file. If you see such error messages, a media or I / O problem has occurred, and should be corrected immediately.

Oracle7 also writes values of initialization parameters to the ALERT file, in addition to other important statistics. For example, when you shutdown an instance normally or immediately (but do not abort), Oracle7 writes the highest number of sessions concurrently connected to the instance, since the instance started, to the ALERT file. You can use this number to see if you need to upgrade your Oracle7 session license.

Specifying the Location of Trace Files

All trace files for background processes and the ALERT file are written to the destination specified by the initialization parameter BACKGROUND_DUMP_DEST. All trace files for server processes are written to the destination specified by the initialization parameter USER_DUMP_DEST. The names of trace files are operating system–specific, but usually include the name of the process writing the file (such as LGWR and RECO).

Controlling the Size of Trace Files

You can control the maximum size of all trace files (excluding the ALERT file) using the initialization parameter MAX_DUMP_FILE_SIZE. This limit is set as a number of operating system blocks. To control the size of an ALERT file, you must manually delete the file when you no longer need it; otherwise Oracle7 continues to append to the file. You can safely delete the ALERT file while the instance is running, although you might want to make an archived copy of it first.

Controlling When Oracle7 Writes to Trace Files

Background processes always write to a trace file when appropriate. However, trace files are written on behalf of server processes (in addition to being written to during internal errors) only if the initialization parameter SQL_TRACE is set to TRUE.

Regardless of the current value of SQL_TRACE, each session can enable or disable trace logging on behalf of the associated server process by using the SQL command ALTER SESSION with the SET SQL_TRACE parameter.

The following statement enables writing to a trace file for a particular session:

```
ALTER SESSION SET SQL_TRACE TRUE;
```

For the multi–threaded server, each session using a dispatcher is routed to a shared server process, and trace information is written to the server's trace file only if the session has enabled tracing (or if an error is encountered). Therefore, to track tracing for a specific session that connects using a dispatcher, you might have to explore several shared server's trace files. Because the SQL trace facility for server processes can cause significant system overhead, enable this feature only when collecting statistics.

See Also: See "Session and User Licensing" on page 19 – 2 for details about upgrading your Oracle license.

For more information about messages, see the *Oracle7 Server Messages* manual.

For information about the names of trace files, see your operating system–specific Oracle documentation.

For complete information about the ALTER SESSION command, see the *Oracle7 Server SQL Reference.*

Starting the Checkpoint Process

If the Checkpoint process (CKPT) is not enabled, the Log Writer process (LGWR) is responsible for updating the headers of all control files and data files to reflect the latest checkpoint. To reduce the time necessary to complete a checkpoint, especially when a database is comprised of many data files, enable the CKPT background process by setting the CHECKPOINT_PROCESS parameter in the database's parameter file to TRUE. (The default is FALSE.)

Managing Processes for the Parallel Query Option

This section describes how, with the parallel query option, Oracle7 can perform parallel processing. In this configuration Oracle7 can divide the work of processing certain types of SQL statements among multiple query server processes. The following topics are included:

- Managing the Query Servers
- Variations in the Number of Query Server Processes

See Also: For more information about the parallel query option, see the *Oracle7 Server Tuning* manual.

Managing the Query Servers

When you start your instance, the Oracle7 Server creates a pool of query server processes available for any query coordinator. Specify the number of query server processes that the Oracle7 Server creates at instance startup via the initialization parameter PARALLEL_MIN_SERVERS.

Query server processes remain associated with a statement throughout its execution phase. When the statement is completely processed, its query server processes become available to process other statements. The query coordinator process returns any resulting data to the user process issuing the statement.

Variations in the Number of Query Server Processes

If the volume of SQL statements processed concurrently by your instance changes drastically, the Oracle7 Server automatically changes the number of query server processes in the pool to accommodate this volume.

If this volume increases, the Oracle7 Server automatically creates additional query server processes to handle incoming statements. The maximum number of query server processes for your instance is specified by the initialization parameter PARALLEL_MAX_SERVERS.

If this volume subsequently decreases, the Oracle7 Server terminates a query server process if it has been idle for the period of time specified by the initialization parameter PARALLEL_SERVER_IDLE_TIME. The Oracle7 Server does not reduce the size of the pool below the value of PARALLEL_MIN_SERVERS, no matter how long the query server processes have been idle.

If all query servers in the pool are occupied and the maximum number of query servers has been started, a query coordinator processes the statement sequentially.

See Also: For more information about monitoring an instance's pool of query servers and determining the appropriate values of the initialization parameters, see the *Oracle7 Server Tuning* manual.

Terminating Sessions

In some situations, you might want to terminate current user sessions. For example, you might want to perform an administrative operation and need to terminate all non–administrative sessions.

This section describes the various aspects of terminating sessions, and includes the following topics:

- Identifying Which Session to Terminate
- Terminating an Active Session
- Terminating an Inactive Session

When a session is terminated, the session's transaction is rolled back and resources (such as locks and memory areas) held by the session are immediately released and available to other sessions.

Terminate a current session using either the Disconnect Session menu item of Server Manager, or the SQL command ALTER SYSTEM...KILL SESSION.

The following statement terminates the session whose SID is 7 and serial number is 15:

```
ALTER SYSTEM KILL SESSION ’7,15’;
```

Identifying Which Session to Terminate

To identify which session to terminate, specify the session's index number and serial number. To identify the index (SID) and serial numbers of a session, query the V$SESSION dynamic performance table.

The following query identifies all sessions for the user JWARD:

```
SELECT sid, serial#
   FROM v$session
   WHERE username = 'JWARD';

SID       SERIAL#    STATUS
--------- ---------- --------
        7         15 ACTIVE
       12         63 INACTIVE
```

A session is ACTIVE when it is making an SQL call to Oracle. A session is INACTIVE if it is not making an SQL call to Oracle.

See Also: For a complete description of the status values for a session, see *Oracle7 Server Tuning.*

Terminating an Active Session

If a user session is making an SQL call to Oracle7 (is ACTIVE) when it is terminated, the transaction is rolled back and the user immediately receives the following message:

```
ORA–00028: your session has been killed
```

If, after receiving the ORA–00028 message, a user submits additional statements before reconnecting to the database, Oracle7 returns the following message:

```
ORA–01012: not logged on
```

If an active session cannot be interrupted (for example, it is performing network I/O or rolling back a transaction), the session cannot be terminated until the operation completes. In this case, the session holds all resources until it is terminated. Additionally, the session that issues the ALTER SYSTEM statement to terminate a session waits up to 60 seconds for the session to be terminated; if the operation that cannot be interrupted continues past one minute, the issuer of the ALTER SYSTEM statement receives a message indicating that the session has been "marked" to be terminated. A session marked to be terminated is indicated in V$SESSION with a status of "KILLED" and a server that is something other than "PSEUDO."

Terminating an Inactive Session

If the session is not making an SQL call to Oracle7 (is INACTIVE) when it is terminated, the ORA–00028 message is not returned immediately. The message is not returned until the user subsequently attempts to use the terminated session.

When an inactive session has been terminated, STATUS in the view V$SESSION is “KILLED.” The row for the terminated session is removed from V$SESSION after the user attempts to use the session again and receives the ORA–00028 message.

In the following example, the DBA terminates an inactive session:

```
SVRMGR> SELECT sid, serial#, status, server
     2>    FROM v$session
     3>    WHERE username = ’JWARD’;

SID        SERIAL# STATUS   SERVER
---------- ------- -------- ---------
         7      15 INACTIVE DEDICATED
        12      63 INACTIVE DEDICATED
2 rows selected.

SVRMGR> ALTER SYSTEM KILL SESSION ’7,15’;
Statement processed.

SVRMGR> SELECT sid, serial#, status, server
     2>    FROM v$session
     3>    WHERE username = ’JWARD’;

SID        SERIAL# STATUS   SERVER
---------- ------- -------- ---------
         7      15 KILLED   PSEUDO
        12      63 INACTIVE DEDICATED
2 rows selected.
```

CHAPTER

5

Managing the Online Redo Log

This chapter explains how to manage the online redo log, and includes the following topics:

- Planning the Online Redo Log
- Creating Online Redo Log Groups and Members
- Renaming and Relocating Online Redo Log Members
- Dropping Online Redo Log Groups
- Dropping Online Redo Log Members
- Controlling Checkpoints and Log Switches
- Verifying Blocks in Redo Log Files
- Clearing an Online Redo Log File
- Listing Information about the Online Redo Log

See Also: For more information about managing the online redo logs of the instances when using Oracle7 Parallel Server, see the *Oracle7 Parallel Server* manual.

For more information archiving the redo log, see Chapter 22.

This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server

Manager / GUI or Server Manager / LineMode, see the *Oracle7 Server Manager User's Guide.*

Planning the Online Redo Log

Every instance of an Oracle database has an associated *online redo log,* which is a set of two or more online log files that record all committed changes made to the database. Online redo logs serve to protect the database in the event of an instance failure. Whenever a transaction is committed, the corresponding redo entries temporarily stored in redo log buffers of the system global area are written to an online redo log file by the background process LGWR.

Online redo log files are used in a cyclical fashion; for example, if two files constitute the online redo log, the first file is filled, the second file is filled, the first file is reused and filled, the second file is reused and filled, and so on. Each time a file is filled, it is assigned a *log sequence number* to identify the set of redo entries.

This section describes guidelines you should consider when configuring a database instance's online redo log, and includes the following topics:

- Multiplexing the Online Redo Log
- Place Online Redo Log Members on Different Disks
- Set the Size of Online Redo Log Members
- Choose an Appropriate Number of Online Redo Log Files

Multiplex the Online Redo Log

The online redo log of a database instance should consist of multiplexed groups of online redo log files. Furthermore, members in the same group should be stored on separate disks so that no single disk failure can cause LGWR and the database instance to fail.

To avoid losing a database due to a single point of failure, Oracle can maintain multiple sets of online redo log files. A *multiplex online redo log* consists of copies of online redo log files physically located on separate disks; changes made to one member of the group are made to all members. If a disk that contains an online redo log file fails, other copies are still intact and available to Oracle. System operation is not interrupted and the lost online redo log files can be easily recovered

⚠ **Warning:** Although the Oracle7 Server allows multiplexed groups to contain different numbers of members, this state should only be temporary, as the result of an abnormal situation such as a disk failure damaging a member of a group. If any group contains only one member, the failure of the disk containing that member could cause Oracle7 to halt.

While multiplexed groups require extra storage space, the cost of this space is usually insignificant compared to the potential cost of lost data (if a disk failure destroys a non–multiplexed online redo log).

Place Online Redo Log Members on Different Disks

With a multiplex online redo log, place members of a group on different disks. This way, if a single disk fails, only one member of a group becomes unavailable to LGWR and other members remain accessible to LGWR, so the instance can continue to function.

If you archive the redo log, spread online redo log members across disks to eliminate contention between the LGWR and ARCH background processes. For example, if you have two groups of duplexed online redo log members, place each member on a different disk and set your archiving destination to a fifth disk. This way, there is never contention between LGWR (writing to the members) and ARCH (reading the members).

Datafiles and online redo log files should also be on different disks to reduce contention in writing data blocks and redo entries.

Set the Size of Online Redo Log Members

When setting the size of online redo log files, consider whether you will be archiving the redo log. Online redo log files should be sized so that a filled group can be archived to a single unit of offline storage media (such as a tape or disk), with the least amount of space on the medium left unused. For example, suppose only one filled online redo log group can fit per tape and 49% of the tape’s storage capacity remains unused. In this case, it would be better to decrease the size of the online redo log files slightly, so that two log groups could be archived per tape.

With multiplex groups of online redo logs, all members of the same group must be the same size. Members of different groups can have different sizes; however, there is no advantage in varying file size between groups. If checkpoints are not set to occur between log switches, make all groups the same size to guarantee that checkpoints occur at regular intervals.

See Also: The default size of online redo log files is operating system–dependent; for more details see your operating system–specific Oracle documentation.

Choose an Appropriate Number of Online Redo Log Files

The best way to determine the appropriate number of online redo log files for a database instance is to test different configurations. The optimum configuration has the fewest groups possible without hampering LGWR's writing redo log information.

In some cases, a database instance may require only two groups. In other situations, a database instance may require additional groups to guarantee that a recycled group is always available to LGWR. During testing, the easiest way to determine if the current online redo log configuration is satisfactory is to examine the contents of the LGWR trace file and the database's ALERT file. If messages indicate that LGWR frequently has to wait for a group because a checkpoint has not completed or a group has not been archived, add groups.

Consider the parameters that can limit the number of online redo log files before setting up or altering the configuration of an instance's online redo log. The following three parameters limit the number of online redo log files that you can add to a database:

- The MAXLOGFILES parameter used in the CREATE DATABASE statement determines the maximum number of groups of online redo log files per database; group values can range from 1 to MAXLOGFILES. The only way to override this upper limit is to re–create the database or its control file; thus, it is important to consider this limit *before* creating a database. If MAXLOGFILES is not specified for the CREATE DATABASE statement, Oracle7 uses an operating system default value.
- The LOG_FILES parameter (in the parameter file) can temporarily decrease the maximum number of groups of online redo log files for the duration of the current instance. However, LOG_FILES cannot override MAXLOGFILES to increase the limit. If LOG_FILES is not set in the database's parameter file, Oracle7 uses an operating system–specific default value.
- The MAXLOGMEMBERS parameter used in the CREATE DATABASE statement determines the maximum number of members per group. As with MAXLOGFILES, the only way to override this upper limit is to re–create the database or control file; thus, it is important to consider this limit *before* creating a database. If no MAXLOGMEMBERS parameter is specified for the CREATE DATABASE statement, Oracle7 uses an operating system default value.

See Also: For the default and legal values of the MAXLOGFILES and MAXLOGMEMBERS parameters, and the LOG_FILES initialization parameter, see your operating system–specific Oracle documentation.

Creating Online Redo Log Groups and Members

You can create groups and members of online redo log files during or after database creation. If you can, plan the online redo log of a database and create all required groups and members of online redo log files during database creation. To create new online redo log groups and members, you must have the ALTER DATABASE system privilege.

In some cases, you might need to create additional groups or members of online redo log files. For example, adding groups to an online redo log can correct redo log group availability problems. A database can have up to MAXLOGFILES groups.

Creating Online Redo Log Groups

To create a new group of online redo log files, use either the Add Logfile Group property sheet of Server Manager, or the SQL command ALTER DATABASE with the ADD LOGFILE parameter.

The following statement adds a new group of redo logs to the database:

```
ALTER DATABASE
   ADD LOGFILE ('log1c', 'log2c') SIZE 500K;
```

> **Note:** Fully specify filenames of new log members to indicate where the operating system file should be created; otherwise, the file is created in the default directory of the database server, which is operating system–dependent. If you want to reuse an existing operating system file, you do not have to indicate the file size.

Using the ALTER DATABASE statement with the ADD LOGFILE option, you can specify the number that identifies the group with the GROUP option:

```
ALTER DATABASE
   ADD LOGFILE GROUP 10 ('log1c', 'log2c') SIZE 500K;
```

Using group numbers can make administering redo log groups easier. However, the group number must be between 1 and MAXLOGFILES; do not skip redo log file group numbers (that is, do not number your groups 10, 20, 30, and so on), or you will consume unnecessary space in the control files of the database.

Creating Online Redo Log Members

In some cases, you might not need to create a complete group of online redo log files; the group may already exist, but not be complete because one or more members of the group were dropped (for example, because of a disk failure). In this case, you can add new members to an existing group.

To create new online redo log members for an existing group, use the Add Logfile Member property sheet of Server Manager, or the SQL command ALTER DATABASE with the ADD LOG MEMBER parameter.

The following statement adds a new redo log member to redo log group number 2:

```
ALTER DATABASE
   ADD LOGFILE MEMBER 'log2b' TO GROUP 2;
```

Notice that filenames must be specified, but sizes need not be; the size of the new members is determined from the size of the existing members of the group.

When using the ALTER DATABASE command, you can alternatively identify the target group by specifying all of the other members of the group in the TO parameter, as shown in the following example:

```
ALTER DATABASE
   ADD LOGFILE MEMBER 'log2c' TO ('log2a', 'log2b');
```

Note: Fully specify the filenames of new log members to indicate where the operating system files should be created; otherwise, the files will be created in the default directory of the database server.

Renaming and Relocating Online Redo Log Members

You can rename online redo log members to change their locations. This procedure is necessary, for example, if the disk currently used for some online redo log files is going to be removed, or if datafiles and a number of online redo log files are stored on the same disk and should be separated to reduce contention.

To rename online redo log members, you must have the ALTER DATABASE system privilege. Additionally, you might also need operating system privileges to copy files to the desired location and privileges to open and back up the database.

Before renaming any online redo log members, ensure that the new online redo log files already exist.

Warning: The following steps only modify the internal file pointers in a database's control files; they do not physically rename or create any operating system files. Use your computer's operating system to copy the existing online redo log files to the new location.

Rename online redo log members with the Rename Logfile Member property sheet of Server Manager, or the SQL command ALTER DATABASE with the RENAME FILE parameter.

To Rename and Relocate Online Redo Log Members

1. Back up the database.

   Before making any structural changes to a database, such as renaming or relocating online redo log members, completely back up the database (including the control file) in case you experience any problems while performing this operation.

2. Copy the online redo log files to the new location.

   Operating system files, such as online redo log members, must be copied using the appropriate operating system commands. See your operating system manual for more information about copying files.

   Suggestion: You can execute an operating system command to copy a file without exiting Server Manager. Use the Server Manager HOST command.

3. Rename the online redo log members.

   Use the Rename Online Redo Log Member dialog box, or the ALTER DATABASE command with the RENAME FILE clause to rename the database's online redo log files.

4. Open the database for normal operation.

   The online redo log alterations take effect the next time that the database is opened. Opening the database may require shutting down the current instance (if the database was previously opened by the current instance) or just opening the database using the current instance.

5. Back up the control file.

   As a precaution, after renaming or relocating a set of online redo log files, immediately back up the database's control file.

The following example renames the online redo log members. However, first assume that:

- The database is currently mounted by, but closed to, the instance.
- The online redo log is duplexed: one group consists of the members LOG1A and LOG1B, and the second group consists of the members LOG2A and LOG2B. The files LOG1A and LOG2A are stored on Disk A, while LOG1B and LOG2B are stored on Disk B.
- The online redo log files located on Disk A must be relocated to Disk C. The new filenames will reflect the new location: LOG1C and LOG2C.

The files LOG1A and LOG2A on Disk A must be copied to the new files LOG1C and LOG2C on Disk C.

```
ALTER DATABASE
   RENAME FILE 'log1a', 'log2a'
      TO 'log1c', 'log2c';
```

Dropping Online Redo Log Groups

In some cases, you might want to drop an entire group of online redo log members. For example, you might want to reduce the number of groups in an instance's online redo log.

To drop an online redo log group, you must have the ALTER DATABASE system privilege.

Before dropping an online redo log group, consider the following restrictions and precautions:

- An instance requires at least two groups of online redo log files, regardless of the number of members in the groups. (A group is one or more members.)
- You can drop an online redo log group only if it is not the active group. If you need to drop the active group, first force a log switch to occur; see “Forcing A Log Switch” on page 5 – 12.

- Make sure an online redo log group is archived (if archiving is enabled) before dropping it. To see whether this has happened, use the Server Manager ARCHIVE LOG command with the LIST parameter.

Drop an online redo log group with either the Drop Logfile Group menu item of Server Manager, or the SQL command ALTER DATABASE with the DROP LOGFILE clause.

The following statement drops redo log group number 3:

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

When an online redo log group is dropped from the database, the operating system files are not deleted from disk. Rather, the control files of the associated database are updated to drop the members of the group from the database structure. After dropping an online redo log group, make sure that the drop completed successfully, and then use the appropriate operating system command to delete the dropped online redo log files.

Dropping Online Redo Log Members

In some cases, you might want to drop one or more specific online redo log members. For example, if a disk failure occurs, you might need to drop all the online redo log files on the failed disk so that Oracle7 does not try to write to the inaccessible files. In other situations, particular online redo log files become unnecessary; for example, a file might be stored in an inappropriate location.

To drop an online redo log member, you must have the ALTER DATABASE system privilege.

Consider the following restrictions and precautions before dropping individual online redo log members:

- It is all right to drop online redo log files so that a multiplexed online redo log becomes temporarily asymmetric. For example, if you use duplexed groups of online redo log files, you can drop one member of one group, even though all other groups have two members each. However, you should rectify this situation immediately so that all groups have at least two members, and thereby eliminate the single point of failure possible for the online redo log.
- An instance always requires at least two valid groups of online redo log files, regardless of the number of members in the groups.

(A group is one or more members.) If the member you want to drop is the last valid member of the group, you cannot drop the member until the other members become valid; to see a redo log file's status, use the V$LOGFILE view. A redo log file becomes INVALID if Oracle7 cannot access it. It becomes STALE if Oracle7 suspects that it is not complete or correct; a stale log file becomes valid again the next time its group is made the active group.

- You can drop an online redo log member only if it is not part of an active group. If you want to drop a member of an active group, first force a log switch to occur.
- Make sure the group to which an online redo log member belongs is archived (if archiving is enabled) before dropping the member. To see whether this has happened, use the Server Manager ARCHIVE LOG command with the LIST parameter.

To drop specific inactive online redo log members, use either the Drop Logfile Member menu item of Server Manager, or the SQL command ALTER DATABASE command with the DROP LOGFILE MEMBER clause.

The following statement drops the redo log LOG3C:

```
ALTER DATABASE   DROP LOGFILE MEMBER 'log3c';
```

When an online redo log member is dropped from the database, the operating system file is not deleted from disk. Rather, the control files of the associated database are updated to drop the member from the database structure. After dropping an online redo log file, make sure that the drop completed successfully, and then use the appropriate operating system command to delete the dropped online redo log file.

See Also: For information on dropping a member of an active group, see "Forcing a Log Switch" on page 5 – 12.

Controlling Checkpoints and Log Switches

A checkpoint is the event during which the Database Writer process (DBWR) writes all modified database buffers in the SGA to the appropriate datafiles. A log switch is the event during which LGWR stops writing to one online redo log group and starts writing to another. The two events are often connected: an instance takes a checkpoint at each log switch by default. A log switch, by default, takes place automatically when the current online redo log file group fills.

However, you can designate that checkpoints are taken more often than when you have log switches, or you can have a checkpoint take place ahead of schedule, without a log switch. You can also have a log switch and checkpoint occur ahead of schedule, or without an accompanying checkpoint.

This section includes the following checkpoint and log switch topics:

- Setting Database Checkpoint Intervals
- Forcing a Log Switch
- Forcing a Fast Database Checkpoint Without a Log Switch

Setting Database Checkpoint Intervals

When your database uses large online redo log files, you can set additional database checkpoints to take place automatically at predetermined intervals, between the checkpoints that automatically occur at log switches. The time necessary to recover from an instance failure decreases when more database checkpoints are set. However, there may be a performance impact on the Oracle7 Server due to the extra I/O necessary for the checkpoint to complete.

Generally, unless your database consistently requires instance recovery on startup, set database checkpoint intervals so that checkpoints occur only at log switches. If you use small online redo log files, checkpoints already occur at frequent intervals (at each log switch).

You can control the frequency of automatic database checkpoints via the values set in the LOG_CHECKPOINT_INTERVAL and LOG_CHECKPOINT_TIMEOUT parameters.

Setting LOG_CHECK–POINT_ INTERVAL

To have database checkpoints only occur at log switches (the default), set the value for the LOG_CHECKPOINT_INTERVAL parameter higher than the size of the online redo log files in use. Alternatively, to force additional checkpoints to occur at intervals between two log switches, set the value for the LOG_CHECKPOINT_INTERVAL parameter lower than the size of the online redo log files in use.

The value of the LOG_CHECKPOINT_INTERVAL is a number of operating system blocks, not Oracle7 data blocks. Therefore, you must know the size, in bytes, of your operating system's blocks. Once you know this, calculate the number of operating system blocks per online redo log file.

As an example, assume the following conditions:

- All online redo log files of the database instance are 512K.
- The operating system block size is 512 bytes.
- Checkpoints should occur when an online redo log file is half full.

Using this information, you can compute the number of blocks per redo log file as follows:

```
512K/redo log file
__________________ = approximately 1000 blocks/redo log file

512 bytes/OS block
```

Now that the approximate number of blocks per online redo log file (1000) is known, the LOG_CHECKPOINT_INTERVAL parameter can be set accordingly in the instance’s parameter file:

```
LOG_CHECKPOINT_INTERVAL=500
```

Setting LOG_CHECK–POINT_TIMEOUT

To have database checkpoints only occur at log switches (the default), set the value for the LOG_CHECKPOINT_TIMEOUT parameter to zero. Alternatively, to force additional checkpoints to occur at intervals between two log switches, set the value for the LOG_CHECKPOINT_TIMEOUT parameter to a time interval (in seconds) less than the average time it takes to fill an online redo log file. To determine the average time it takes to fill online redo log files, examine the LGWR trace file for messages that indicate the times of log switches.

See Also: For information on how to determine operating system block size, see your operating system–specific Oracle documentation.

For more information about tuning Oracle7 regarding checkpoints, see the *Oracle7 Server Tuning* manual.

For more information about the LOG_CHECKPOINT_TIMEOUT parameter when using the Oracle7 Parallel Server, see the *Oracle7 Parallel Server Concepts & Administration* manual.

For more information about setting LOG_CHECKPOINT_TIMEOUT when using Trusted Oracle7 in OS MAC mode, see the *Trusted Oracle7 Server Administrator’s Guide.*

Forcing a Log Switch

You can force a log switch to make the currently active group inactive and available for online redo log maintenance operations. For example, you want to drop the currently active group, but are not able to do so until the group is inactive. You may also wish to force a log switch if the currently active group needs to be archived at a specific time before the members of the group are completely filled; this option is often useful in configurations with large online redo log files that take a long time to fill.

To force a log switch, you must have the Alter System privilege.To force a log switch, use either the Switch Logfile menu item of Server Manager, or the SQL command ALTER SYSTEM with the SWITCH LOGFILE option.

The following statement forces a log switch:

```
ALTER SYSTEM SWITCH LOGFILE;
```

Forcing a Fast Database Checkpoint Without a Log Switch

In some cases, you might want to force a fast database checkpoint. A fast checkpoint is one which does not involve a log switch; LGWR continues to write to the current online redo log file. A fast checkpoint allows DBWR to write more modified database buffers to disk per I/O on behalf of a checkpoint. Therefore, you need fewer I/O's (thus less time) to complete a fast checkpoint.

To force a database checkpoint, you must have the ALTER SYSTEM system privilege. Force a fast database checkpoint with either the Force Checkpoint menu item of Server Manager, or the SQL command ALTER SYSTEM with the CHECKPOINT option.

The following statement forces a checkpoint:

```
ALTER SYSTEM CHECKPOINT;
```

Omitting the GLOBAL option allows you to force a checkpoint for only the connected instance, while including it forces a checkpoint for all instances of the database. Forcing a checkpoint for only the local instance is useful only with the Oracle7 Parallel Server. In a non–parallel server configuration, global and local checkpoints are identical.

See Also: For more information on forcing checkpoints with the Oracle7 Parallel Server, see the *Oracle7 Parallel Server Concepts & Administration* manual.

Verifying Blocks in Redo Log Files

You can configure Oracle7 to use checksums to verify blocks in the redo log files. Set the initialization parameter LOG_BLOCK_CHECKSUM to TRUE to enable redo log block checking. The default value of LOG_BLOCK_CHECKSUM is FALSE.

If you enable redo log block checking, Oracle7 computes a checksum for each redo log block written to the current log. The checksums are written in the header of the block.

Oracle7 uses the checksum to detect corruption in a redo log block. Oracle7 tries to verify the redo log block when it writes the block to an archive log file and when the block is read from an archived log during recovery.

If Oracle7 detects a corruption in a redo log block while trying to archive it, Oracle7 tries to read the block from another member in the group. If the block is corrupted in all members the redo log group, then archiving cannot proceed.

See Also: For information about archiving redo log files, see Chapter 22.

Clearing an Online Redo Log File

If you have enabled redo log block checking, Oracle7 verifies each block before archiving it. If a particular redo log block is corrupted in all members of a group, archiving stops. Eventually all the redo logs become filled and database activity is halted, until archiving can resume.

In this situation, you can use the SQL command ALTER DATABASE... CLEAR LOGFILE to clear the corrupted redo logs and avoid archiving them. The cleared redo logs are available for use even though they were not archived.

The following statement clears the log files in redo log group number 3:

```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE
   GROUP 3;
```

Restrictions

You can clear a redo log file whether it is archived or not. However, when it is not archived, you must include the keyword UNARCHIVED.

If you clear a log file that is needed for recovery of a backup, then you can no longer recover from that backup. Oracle7 writes a message in the alert log describing the backups from which you cannot recover.

☞ **Attention:** If you clear an unarchived redo log file, you should take another backup of the database.

If you want to clear an unarchived redo log that is needed to bring an offline tablespace online, you must use the clause UNRECOVERABLE DATAFILE in the ALTER DATABASE command.

If you clear a redo log needed to bring an offline tablespace online, you will not be able to bring the tablespace online again. You will have to drop the tablespace or perform an incomplete recovery.

See Also: For a complete description of the ALTER DATABASE command, see the *Oracle7 Server SQL Reference.*

Listing Information about the Online Redo Log

Use the VLOG, VLOGFILE, and V$THREAD views to see information about the online redo log of a database; the V$THREAD view is of particular interest for Parallel Server administrators.

The following query returns information about the online redo log of a database used without the Parallel Server:

```
SELECT group#, bytes, members
   FROM sys.v$log;

GROUP#     BYTES      MEMBERS
---------- ---------- ----------
         1      81920          2
         2      81920          2
```

To see the names of all of the member of a group, use a query similar to the following:

```
SELECT *
   FROM sys.v$logfile
   WHERE group# = 2;

GROUP#     STATUS      MEMBER
---------- ----------- --------------
         2             LOG2A
         2 STALE       LOG2B
         2             LOG2C
```

If STATUS is blank for a member, the file is in use.

CHAPTER

6

Managing Control Files

This chapter explains how to create and maintain the control files for your database, and includes the following topics:

- Guidelines for Control Files
- Creating Control Files
- Troubleshooting After Creating Control Files
- Dropping Control Files

See Also: This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle7 Server Manager User's Guide*.

Guidelines for Control Files

This section describes guidelines you can use to manage the control files for a database, and includes the following topics:

- Name Control Files
- Multiplex Control Files on Different Disks
- Place Control Files Appropriately
- Manage the Size of Control Files

Name Control Files

Assign control file names via the CONTROL_FILES initialization parameter in the database's parameter file. CONTROL_FILES indicates one or more names of control files, separated by commas. The instance startup procedure recognizes and opens all the listed files. The instance maintains all listed control files during database operation.

During database operation, Oracle7 Server writes to all necessary files listed for the CONTROL_FILES parameter.

Multiplex Control Files on Different Disks

Every Oracle7 database should have at least two control files, each stored on a different disk. If a control file is damaged due to a disk failure, the associated instance must be shut down. Once the disk drive is repaired, the damaged control file can be restored using an intact copy of the control file and the instance can be restarted; no media recovery is required.

Behavior of Multiplexed Control Files

The following list describes the behavior of multiplexed control files:

- Two or more filenames are listed for the initialization parameter CONTROL_FILES in the database's parameter file.
- The first file listed in the CONTROL_FILES parameter is the only file read by the Oracle7 Server during database operation.
- If any of the control files become unavailable during database operation, the instance becomes inoperable and should be aborted.

The only disadvantage of having multiple control files is that all operations that update the control files (such as adding a datafile or checkpointing the database) can take slightly longer. However, this difference is usually insignificant (especially for operating systems that can perform multiple, concurrent writes) and does not justify using only a single control file.

☞ **Attention:** Oracle strongly recommends that your database has a minimum of two control files on different disks**.**

Place Control Files Appropriately

Each copy of a control file should be stored on a different disk drive. Furthermore, a control file copy should be stored on every disk drive that stores members of online redo log groups, if the online redo log is multiplexed. By storing control files in these locations, you minimize the risk that all control files and all groups of the online redo log will be lost in a single disk failure.

Manage the Size of Control Files

The main determinants of a control file's size are the values set for the MAXDATAFILES, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, and MAXINSTANCES parameters in the CREATE DATABASE statement that created the associated database. Increasing the values of these parameters increases the size of a control file of the associated database.

See Also: The maximum control file size is operating system–specific. See your operating system–specific Oracle documentation for more information.

Creating Control Files

Every Oracle7 database has a *control file.* A control files records the physical structure of the database and contains:

- the database name
- names and locations of associated databases and online redo log files
- the timestamp of the database creation
- the current log sequence number
- checkpoint information

The control file of an Oracle7 database is created at the same time as the database. By default, at least one copy of the control file must be created during database creation. On some operating systems, Oracle7 creates multiple copies. You should create two or more copies of the control file during database creation. You might also need to create control files later, if you lose control files or want to change particular settings in the control files.

This section describes ways to create control files, and includes the following topics:

- Creating Initial Control Files

- Creating Additional Copies of the Control File, and Renaming and Relocating Control Files
- New Control Files
- Creating New Control Files

Creating Initial Control Files

You create the initial control files of an Oracle7 database by specifying one or more control filenames in the CONTROL_FILES parameter in the parameter file used during database creation. The filenames specified in CONTROL_FILES should be fully specified. Filename specification is operating system–specific.

If files with the specified names currently exist at the time of database creation, you must specify the CONTROLFILE REUSE parameter in the CREATE DATABASE command, or else an error occurs. Also, if the size of the old control file differs from that of the new one, you cannot use the REUSE option. The size of the control file changes between some release of new version of Oracle, as well as when the number of files specified in the control file changes; configuration parameters such as MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES affect control file size.

If you do not specify files for CONTROL_FILES before database creation, Oracle7 uses a default filename. The default name is also operating system–specific.

You can subsequently change the value of the CONTROL_FILES parameter to add more control files or to change the names or locations of existing control files.

See Also: For more information about specifying control files, see your operating system–specific Oracle documentation.

Creating Additional Copies of the Control File, and Renaming and Relocating Control Files

You add a new control file by copying an existing file to a new location and adding the file's name to the list of control files.

Similarly, you rename an existing control file by copying the file to its new name or location, and changing the file's name in the control file list.

In both cases, to guarantee that control files do not change during the procedure, shut down the instance before copying the control file.

To Multiplex or Move Additional Copies of the Current Control File

1. Shutdown the database.
2. Exit Server Manager.
3. Copy an existing control file to a different location, using operating system commands.
4. Edit the CONTROL_FILES parameter in the database's parameter file to add the new control file's name, or to change the existing control filename.
5. Restart Server Manager.
6. Restart the database.

New Control Files

You can create a new control file for a database using the CREATE CONTROLFILE command. This is recommended in the following situations:

- All control files for the database have been permanently damaged and you do not have a control file backup.
- You want to change one of the permanent database settings originally specified in the CREATE DATABASE statement, including the database's name, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES.

  For example, you might need to change a database's name if it conflicts with another database's name in a distributed environment. As another example, you might need to change one of the previously mentioned parameters if the original setting is too low.

The following statement creates a new control file for the PROD database (formerly a database that used a different database name):

```
CREATE CONTROLFILE
    SET DATABASE prod
    LOGFILE GROUP 1 ('logfile1A', 'logfile1B') SIZE 50K,
       GROUP 2 ('logfile2A', 'logfile2B') SIZE 50K
    NORESETLOGS
    DATAFILE 'datafile1' SIZE 3M, 'datafile2' SIZE 5M
    MAXLOGFILES 50
    MAXLOGMEMBERS 3
    MAXDATAFILES 200
    MAXINSTANCES 6
    ARCHIVELOG;
```

⚠ **Warning:** The CREATE CONTROLFILE command can potentially damage specified datafiles and online redo log files; omitting a filename can cause loss of the data in that file, or loss of access to the entire database. Employ caution when using this command and be sure to follow the steps in the next section.

See Also: For more information about the CREATE CONTROLFILE command, see the *Oracle7 Server SQL Reference.*

Creating New Control Files

This section provides step–by–step instructions for creating new control files.

To Create New Control Files

1. Make a list of all datafiles and online redo log files of the database.

   If you followed the recommendations for database backups, you should already have a list of datafiles and online redo log files that reflect the current structure of the database.

   If you have no such lists and your control file has been damaged so that the database cannot be opened, try to locate all of the datafiles and online redo log files that constitute the database. Any files not specified in Step 5 are not recoverable once a new control file has been created. Moreover, if you omit any of the files that make up the SYSTEM tablespace, you might not be able to recover the database.

2. Shut down the database.

   If the database is open, shut down the database with normal priority, if possible. Use the immediate or abort options only as a last resort.

3. Back up all datafiles and online redo log files of the database.

4. Start up an new instance, but do not mount or open the database.

5. Create a new control file for the database using the CREATE CONTROLFILE command.

   When creating the new control file, select the RESETLOGS option if you have lost any online redo log groups in addition to the control files. In this case, you will need to recover from the loss of the redo logs (Step 8). You must also specify the RESETLOGS option if you have renamed the database. Otherwise, select the NORESETLOGS option.

6. Store a backup of the new control file on an offline storage device.

7. Edit the parameter files of the database.

   Edit the parameter files of the database to indicate all of the control files created in Steps 5 and 6 (not including the backup control file) in the CONTROL_FILES parameter.

8. Recover the database if necessary.

   If you are creating the control file as part of recovery, recover the database. If the new control file was created using the NORESETLOGS option (Step 5), you can recover the database with complete, closed database recovery.

   If the new control file was created using the RESETLOGS option, you must specify USING BACKUP CONTROL FILE . If you have lost online or archived redo logs or datafiles, use the procedures for recovering those files.

9. Open the database.

   Open the database using one of the following methods:

   - If you did not perform recovery, open the database normally.
   - If you performed complete, closed database recovery in Step 8, use the Startup Open radio button of the Startup Database dialog box of Server Manager.
   - If you specified RESETLOGS when creating the control file, use the ALTER DATABASE command, indicating RESETLOGS.

The database is now open and available for use.

See Also: For more information about listing database files, see “Listing Database Files Before Backup” on page 23 – 8.

For more information on backing up all datafiles and online redo log files of the database, see “Performing a Full Backup” on page 23 – 9.

For more information on recovering online or archived redo log files, see "Loss of Online Redo Log Files" on page 24 – 49 and "Loss of Datafiles" on page 24 – 48.

For more information on closed database recovery, see page 24 – 18.

Troubleshooting After Creating Control Files

After issuing the CREATE CONTROLFILE statement, you may encounter some common errors. This section describes the most common control file usage errors, and includes the following topics:

- Checking for Missing or Extra Files
- Handling Errors During CREATE CONTROLFILE

Checking for Missing or Extra Files

After creating a new control file and using it to open the database, check the ALERT log to see if Oracle7 has detected inconsistencies between the data dictionary and the control file, such as a datafile that the data dictionary includes but the control file does not list.

If a datafile exists in the data dictionary but not in the new control file, Oracle7 creates a placeholder entry in the control file under the name MISSING*nnnn* (where *nnnn* is the file number in decimal). MISSING*nnnn* is flagged in the control file as being offline and requiring media recovery.

In the following two cases only, the actual datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn* to point to it.

Case 1: The new control file was created using the CREATE CONTROLFILE command with the NORESETLOGS option, thus allowing the database to be opened without using the RESETLOGS option. This would be possible only if all online redo logs are available.

Case 2: It was necessary to use the RESETLOGS option on the CREATE CONTROLFILE command, thus forcing the database to be opened using the RESETLOGS option, but the actual datafile corresponding to MISSING*nnnn* was read–only or offline normal.

If, on the other hand, it was necessary to open the database using the RESETLOGS option, and MISSING*nnnn* corresponds to a datafile that was not read–only or offline normal, then the rename operation cannot be used to make the datafile accessible (since the datafile requires media recovery that is precluded by the results of RESETLOGS). In this case, the tablespace containing the datafile must be dropped.

In contrast, if a datafile indicated in the control file is not present in the data dictionary, Oracle7 removes references to it from the new control file. In both cases, Oracle7 includes an explanatory message in the ALERT file to let you know what it found.

Handling Errors During CREATE CONTROLFILE

If Oracle7 sends you an error (usually error ORA–01173, ORA–01176, ORA–01177, ORA–01215, or ORA–01216) when you attempt to mount and open the database after creating a new control file, the most likely cause is that you omitted a file from the CREATE CONTROLFILE statement or included one that should not have been listed. In this case, you should restore the files you backed up in step 3 and repeat the procedure from step 4, using the correct filenames.

Dropping Control Files

You can drop control files from the database. For example, you might want to do so if the location of a control file is inappropriate. Remember that the database must have at least two control files at all times.

To Drop a Control File from a Database

1. Shut down the database.
2. Exit Server Manager.
3. Edit the CONTROL_FILES parameter in the database’s parameter file to delete the old control file’s name.
4. Restart Server Manager.
5. Restart the database.

⚠ **Warning:** This operation does not physically delete the unwanted control file from the disk. Use operating system commands to delete the unnecessary file after you have dropped the control file from the database.

CHAPTER

7 Managing Job Queues

This chapter describes how to use job queues to schedule periodic execution of PL/SQL code, and includes the following topics:

- SNP Background Processes
- Managing Job Queues
- Viewing Job Queue Information

See Also: This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager/GUI or Server Manager/LineMode, see the *Oracle Server Manager User's Guide.*

SNP Background Processes

This section describes SNP background processes and their role in managing job queues, and includes the following topics:

- Multiple SNP Processes
- Starting SNP Processes

You can schedule routines to be performed periodically using the job queue. A routine is any PL/SQL code. To schedule a job, you submit it to the job queue and specify the frequency at which the job is to be run. You can also alter, disable, or delete jobs you have submitted.

To maximize performance and accommodate many users, a multi–process Oracle7 system uses some additional processes called *background processes*. Background processes consolidate functions that would otherwise be handled by multiple Oracle programs running for each user process. Background processes asynchronously perform I/O and monitor other Oracle processes to provide increased parallelism for better performance and reliability.

SNP background processes execute job queues. SNP processes periodically wake up and execute any queued jobs that are due to be run. You must have at least one SNP process running to execute your queued jobs in the background.

SNP background processes differ from other Oracle7 background processes, in that the failure of an SNP process does not cause the instance to fail. If an SNP process fails, Oracle7 restarts it.

See Also: For more information on background processes, see *Oracle7 Server Concepts*.

Multiple SNP Processes

An instance can have up to ten SNP processes, named SNP0 to SNP9. If an instance has multiple SNP processes, the task of executing queued jobs can be shared across these processes, thus improving performance. Note, however, that each job is run at any point in time by only one process. A single job cannot be shared simultaneously by multiple SNP processes.

Starting Up SNP Processes

Job queue initialization parameters enable you to control the operation of the SNP background processes. When you set these parameters in the initialization parameter file for an instance, they take effect the next time you start the instance.

Table 7 – 1 describes the job queue initialization parameters.

Parameter Name	Description
JOB_QUEUE_PROCESSES	Default: 0 Range of values: 0...10 Multiple instances: can have different values Sets the number of SNP background processes per instance.
JOB_QUEUE_INTERVAL	Default: 60 (seconds) Range of values: 1...3600 (seconds) Multiple instances: can have different values Sets the interval between wake–ups for the SNP background processes of the instance.

Table 7 – 1 Job Queue Initialization Parameters

Managing Job Queues

This section describes the various aspects of managing job queues, and includes the following topics:

- DBMS_JOB Package
- Submitting a Job to the Job Queue
- How Jobs Execute
- Removing a Job From the Job Queue
- Altering a Job
- Broken Jobs
- Forcing a Job to Execute
- Terminating a Job

DBMS_JOB Package

To schedule and manage jobs in the job queue, use the procedures in the DBMS_JOB package. There are no database privileges associated with using job queues. Any user who can execute the job queue procedures can use the job queue. Table 7 – 2 lists the job queue procedures in the DBMS_JOB package.

Procedure	Description	Described on
SUBMIT	Submits a job to the job queue.	page 7 – 4
REMOVE	Removes specified job from the job queue.	page 7 – 9
CHANGE	Alters a specified job. You can alter the job description, the time at which the job will be run, or the interval between executions of the job.	page 7 – 10
WHAT	Alters the job description for a specified job.	page 7 – 10
NEXT_DATE	Alters the next execution time for a specified job.	page 7 – 11
INTERVAL	Alters the interval between executions for a specified job.	page 7 – 11
BROKEN	Disables job execution. If a job is marked as broken, Oracle7 does not attempt to execute it.	page 7 – 11
RUN	Forces a specified job to run.	page 7 – 12

Table 7 – 2 Procedures in the DBMS_JOB Package

Submitting a Job to the Job Queue

To submit a new job to the job queue, use the SUBMIT procedure in the DBMS_JOB package:

```
DBMS_JOB.SUBMIT(   job            OUT    BINARY_INTEGER,
                   what           IN     VARCHAR2,
                   next_date      IN     DATE DEFAULT SYSDATE,
                   interval       IN     VARCHAR2 DEFAULT ’null’,
                   no_parse       IN     BOOLEAN DEFAULT FALSE)
```

The SUBMIT procedure returns the number of the job you submitted. Table 7 – 3 describes the procedure’s parameters.

Parameter	Description
`job`	This is the identifier assigned to the job you created. You must use the job number whenever you want to alter or remove the job. For more information about job numbers, see “Job Numbers” on page 7 – 6.
`what`	This is the PL/SQL code you want to have executed. For more information about defining a job, see “Job Definitions” on page 7 – 7.
`next_date`	This is the next date when the job will be run. The default value is SYSDATE.
`interval`	This is the date function that calculates the next time to execute the job. The default value is NULL. INTERVAL must evaluate to a future point in time or NULL. For more information on how to specify an execution interval, see page 7 – 7.
`no_parse`	This is a flag. The default value is FALSE. If NO_PARSE is set to FALSE (the default), Oracle7 parses the procedure associated with the job. If NO_PARSE is set to TRUE, Oracle7 parses the procedure associated with the job the first time that the job is executed. If, for example, you want to submit a job before you have created the tables associated with the job, set NO_PARSE to TRUE.

Table 7 – 3 Parameters for DBMS_JOB.SUBMIT

As an example, let’s submit a new job to the job queue. The job calls the procedure DBMS_DDL.ANALYZE_OBJECT to generate optimizer statistics for the table DQUON.ACCOUNTS. The statistics are based on a sample of half the rows of the ACCOUNTS table. The job is run every 24 hours:

```
SVRMGR> VARIABLE jobno number;
SVRMGR> begin
     2>           DBMS_JOB.SUBMIT(:jobno,
     3>                  ’dbms_ddl.analyze_object(’’TABLE’’,
     4>                  ’’DQUON’’, ’’ACCOUNTS’’,
     5>                  ’’ESTIMATE’’, NULL, 50);’
     6>                  SYSDATE, ’SYSDATE + 1’);
     7> end;
     8> /
Statement processed.
SVRMGR> print jobno
JOBNO
----------
     14144
```

Job Environment

When you submit a job to the job queue or alter a job's definition, Oracle7 records the following environment characteristics:

- the current user
- the user submitting or altering a job
- the current schema
- MAC privileges (if appropriate)

Oracle also records the following NLS parameters:

- NLS_LANGUAGE
- NLS_TERRITORY
- NLS_CURRENCY
- NLS_ISO_CURRENCY
- NLS_NUMERIC_CHARACTERS
- NLS_DATE_FORMAT
- NLS_DATE_LANGUAGE
- NLS_SORT

Oracle restores these environment characteristics every time a job is executed. NLS_LANGUAGE and NLS_TERRITORY parameters are defaults for unspecified NLS parameters.

You can change a job's environment by using the DBMS_SQL package and the ALTER SESSION command.

Jobs and Import/Export

Jobs can be exported and imported. Thus, if you define a job in one database, you can transfer it to another database. When exporting and importing jobs, the job's number, environment, and definition remain unchanged.

> **Note:** If the job number of a job you want to import matches the number of a job already existing in the database, you will not be allowed to import that job. Submit the job as a new job in the database.

Job Owners

When you submit a job to the job queue, Oracle7 identifies you as the owner of the job. Only a job's owner can alter the job, force the job to run, or remove the job from the queue.

Job Numbers

A queued job is identified by its job number. When you submit a job, its job number is automatically generated from the sequence SYS.JOBSEQ.

Once a job is assigned a job number, that number does not change. Even if the job is exported and imported, its job number remains the same.

Job Definitions

The *job definition* is the PL/SQL code specified in the WHAT parameter of the SUBMIT procedure.

Normally the job definition is a single call to a procedure. The procedure call can have any number of parameters.

> **Note:** In the job definition, use two single quotation marks around strings. Always include a semicolon at the end of the job definition.

There are special parameter values that Oracle7 recognizes in a job definition. Table 7 – 4 lists these parameters.

Parameter	Mode	Description
`job`	IN	The number of the current job.
`next_date`	IN/OUT	The date of the next execution of the job. The default value is SYSDATE.
`broken`	IN/OUT	Status of job, broken or not broken. The IN value is FALSE.

Table 7 – 4 Special Parameter Values for Job Definitions

The following are examples of valid job definitions:

```
'myproc(''10–JAN–82'', next_date, broken);'
'scott.emppackage.give_raise(''JFEE'', 3000.00);'
'dbms_job.remove(job);'
```

Job Execution Interval

The INTERVAL date function is evaluated immediately before a job is executed. If the job completes successfully, the date calculated from INTERVAL becomes the new NEXT_DATE. If the INTERVAL date function evaluates to NULL and the job completes successfully, the job is deleted from the queue.

If a job should be executed periodically at a set interval, use a date expression similar to 'SYSDATE + 7' in the INTERVAL parameter. For example, if you set the execution interval to 'SYSDATE + 7' on Monday, but for some reason (such as a network failure) the job is not executed until Thursday, 'SYSDATE + 7' then executes every Thursday, not Monday.

If you always want to automatically execute a job at a specific time, regardless of the last execution (for example, every Monday), the INTERVAL and NEXT_DATE parameters should specify a date expression similar to 'NEXT_DAY(TRUNC(SYSDATE), ''MONDAY'')'.

Table 7 – 5 lists some common date expressions used for job execution intervals.

Date Expression	Evaluation
'SYSDATE + 7'	exactly seven days from the last execution
'SYSDATE + 1/48'	every half hour
'NEXT_DAY(TRUNC(SYSDATE), ''MONDAY'') + 15/24'	every Monday at 3PM
'NEXT_DAY(ADD_MONTHS (TRUNC(SYSDATE, ''Q''), 3), ''THURSDAY'')'	first Thursday of each quarter

Table 7 – 5 Common Job Execution Intervals

> **Note:** When specifying NEXT_DATE or INTERVAL, remember that date literals and strings must be enclosed in single quotation marks. Also, the value of INTERVAL must be enclosed in single quotation marks.

Database Links and Jobs

If you submit a job that uses a database link, the link must include a username and password. Anonymous database links will not succeed.

See Also: For more information about the ALTER SESSION command, see *Oracle7 Server SQL Reference.*

For more information on the DBMS_SQL package, see the *Oracle7 Server Application Developer's Guide.*

How Jobs Execute

SNP background processes execute jobs. To execute a job, the process creates a session to run the job.

When an SNP process runs a job, the job is run in the same environment in which it was submitted and with the owner's default privileges.

When you force a job to run using the procedure DBMS_JOB.RUN, the job is run by your user process. When your user process runs a job, it is run with your default privileges only. Privileges granted to you through roles are unavailable.

Job Queue Locks

Oracle7 uses job queue locks to ensure that a job is executed one session at a time. When a job is being run, its session acquires a job queue (JQ) lock for that job.

Interpreting Information about JQ Locks You can use the Server Manager Lock Monitor or the locking views in the data dictionary to examine information about locks currently held by sessions.

The following query lists the session identifier, lock type, and lock identifiers for all sessions holding JQ locks:

```
SVRMGR> SELECT sid, type, id1, id2
     2>    FROM v$lock
     3>    WHERE type = ’JQ’;

SID        TY ID1        ID2
---------- -- ---------- ----------
        12 JQ          0      14144
1 row selected.
```

In the query above, the identifier for the session holding the lock is 12. The ID1 lock identifier is always 0 for JQ locks. The ID2 lock identifier is the job number of the job the session is running.

Job Execution Errors

When a job fails, information about the failure is recorded in a trace file and the alert log. Oracle7 writes message number ORA–12012 and includes the job number of the failed job.

The following can prevent the successful execution of queued jobs:

- not having any SNP background processes to run the job
- a network or instance failure
- an exception when executing the job

Job Failure and Execution Times If a job returns an error while Oracle7 is attempting to execute it, Oracle7 tries to execute it again. The first attempt is made after one minute, the second attempt after two minutes, the third after four minutes, and so on, with the interval doubling between each attempt. When the retry interval exceeds the execution interval, Oracle7 continues to retry the job at the normal execution interval. However, if the job fails sixteen times, Oracle7 automatically marks the job as broken and no longer tries to execute it.

Thus, if you can correct the problem that is preventing a job from running before the job has failed sixteen times, Oracle7 will eventually run that job again.

See Also: For more information about the locking views, see the *Oracle7 Server Reference.*

For more information about locking, see *Oracle7 Server Concepts.*

Removing a Job From the Job Queue

To remove a job from the job queue, use the REMOVE procedure in the DBMS_JOB package:

```
DBMS_JOB.REMOVE(job IN BINARY_INTEGER)
```

The following statement removes job number 14144 from the job queue:

```
DBMS_JOB.REMOVE(14144);
```

Restrictions

You can remove currently executing jobs from the job queue. However, the job will not be interrupted, and the current execution will be completed.

You can only remove jobs you own. If you try to remove a job that you do not own, you receive a message that states the job is not in the job queue.

Altering a Job

To alter a job that has been submitted to the job queue, use the procedures CHANGE, WHAT, NEXT_DATE, or INTERVAL in the DBMS_JOB package.

Here's an example where the job identified as 14144 is now executed every three days:

```
DBMS_JOB.CHANGE(14144, null, null, 'SYSDATE + 3');
```

Restrictions

You can only alter jobs that you own. If you try to alter a job that you do not own, you receive a message that states the job is not in the job queue.

Syntax for CHANGE

You can alter any of the user–definable parameters associated with a job by calling the DBMS_JOB.CHANGE procedure. Table 7 – 3 describes the procedure's parameters:

```
DBMS_JOB.CHANGE(   job            IN      BINARY_INTEGER,
                   what           IN      VARCHAR2,
                   next_date      IN      DATE,
                   interval       IN      VARCHAR2)
```

If you specify NULL for WHAT, NEXT_DATE, or INTERVAL when you call the procedure CHANGE, the current value remains unchanged.

> **Note:** When you change a job's definition using the WHAT parameter in the procedure CHANGE, Oracle7 records your current environment. This becomes the new environment for the job.

Syntax for WHAT

You can alter the definition of a job by calling the DBMS_JOB.WHAT procedure. Table 7 – 3 describes the procedure's parameters:

```
DBMS_JOB.WHAT(  job               IN      BINARY_INTEGER,
                what              IN      VARCHAR2)
```

> **Note:** When you execute procedure WHAT, Oracle7 records your current environment. This becomes the new environment for the job.

Syntax for NEXT_DATE

You can alter the next date that Oracle7 executes a job by calling the DBMS_JOB.NEXT_DATE procedure. Table 7 – 3 describes the procedure's parameters:

```
DBMS_JOB.NEXT_DATE( job                 IN      BINARY_INTEGER,
                    next_date           IN      DATE)
```

Syntax for INTERVAL

You can alter the execution interval of a job by calling the DBMS_JOB.INTERVAL procedure. Table 7 – 3 describes the procedure's parameters:

```
DBMS_JOB.INTERVAL( job            IN      BINARY_INTEGER,
                   interval       IN      VARCHAR2)
```

Broken Jobs

A job is labeled as either broken or not broken. Oracle7 does not attempt to run broken jobs. However, you can force a broken job to run by calling the procedure DBMS_JOB.RUN.

When you submit a job it is considered not broken.

There are two ways a job can break:

- Oracle7 has failed to successfully execute the job after sixteen attempts.
- You have marked the job as broken, using the procedure DBMS_JOB.BROKEN.

To mark a job as broken or not broken, use the procedure BROKEN in the DBMS_JOB package. Table 7 – 4 describes the procedure's parameters:

```
DBMS_JOB.BROKEN(   job            IN      BINARY_INTEGER,
                   broken         IN      BOOLEAN,
                   next_date      IN      DATE DEFAULT SYSDATE)
```

The following example marks job 14144 as not broken and sets its next execution date to the following Monday:

```
DBMS_JOB.BROKEN(14144, FALSE, NEXT_DAY(SYSDATE, 'MONDAY'));
```

Once a job has been marked as broken, Oracle7 will not attempt to execute the job until you either mark the job as not broken, or force the job to be executed by calling the procedure DBMS_JOB.RUN.

Restrictions

You can only mark jobs you own as broken. If you try to mark a job you do not own, you receive a message that states the job is not in the job queue.

Running Broken Jobs

If a problem has caused a job to fail sixteen times, Oracle7 marks the job as broken. Once you have fixed this problem, you can run the job by either:

- forcing the job to run by calling DBMS_JOB.RUN
- marking the job as not broken by calling DBMS_JOB.BROKEN and waiting for Oracle7 to execute the job

If you force the job to run by calling the procedure DBMS_JOB.RUN, Oracle7 runs the job immediately. If the job succeeds, then Oracle7 labels the job as not broken and resets its count of the number of failed executions for the job.

Once you reset a job's broken flag (by calling either RUN or BROKEN), job execution resumes according to the scheduled execution intervals set for the job.

Forcing a Job to Execute

There may be times when you would like to manually execute a job. For example, if you have fixed a broken job, you may want to test the job immediately by forcing it to execute.

To force a job to be executed immediately, use the procedure RUN in the DBMS_JOB package. Oracle7 attempts to run the job, even if the job is marked as broken:

```
DBMS_JOB.RUN(      job     IN      BINARY_INTEGER)
```

When you run a job using DBMS_JOB.RUN, Oracle7 recomputes the next execution date. For example, if you create a job on a Monday with a NEXT_DATE value of 'SYSDATE' and an INTERVAL value of 'SYSDATE + 7', the job is run every 7 days starting on Monday. However, if you execute RUN on Wednesday, the next execution date will be the next Wednesday.

> **Note:** When you force a job to run, the job is executed in your current session. Running the job reinitializes your session's packages.

Restrictions

You can only run jobs that you own. If you try to run a job that you do not own, you receive a message that states the job is not in the job queue.

The following statement runs job 14144 in your session and recomputes the next execution date:

```
DBMS_JOB.RUN(14144);
```

The procedure RUN contains an implicit commit. Once you execute a job using RUN, you cannot rollback.

Terminating a Job

You can terminate a running job by marking the job as broken, identifying the session running the job, and disconnecting that session. You should mark the job as broken, so that Oracle7 does not attempt to run the job again.

After you have identified the session running the job (via V$SESSION), you can disconnect the session using the Server Manager Disconnect Session menu item, or the SQL command ALTER SYSTEM.

See Also: For examples of viewing information about jobs and sessions, see "Viewing Job Queue Information" on page 7 – 13.

For more information on V$SESSION, see the *Oracle7 Server Reference*.

Viewing Job Queue Information

You can view information about jobs in the job queue via the data dictionary views in Table 7 – 6:

View	Description
DBA_JOBS	Lists all the jobs in the database.
USER_JOBS	Lists all jobs owned by the user.
DBA_JOBS_RUNNING	Lists all jobs in the database that are currently running. This view joins V$LOCK and JOB$.

Table 7 – 6 Views for Viewing Job Queue Information

For example, you can display information about a job's status and failed executions. The following sample query creates a listing of the job number, next execution time, failures, and broken status for each job you have submitted:

```
SVRMGR> SELECT job, next_date, next_sec, failures, broken
     2>    FROM user_jobs;

JOB        NEXT_DATE NEXT_SEC FAILURES   B
---------- --------- -------- ---------- -
      9125 01–NOV–94 00:00:00          4 N
     14144 24–OCT–94 16:35:35          0 N
     41762 01–JAN–00 00:00:00         16 Y
3 rows selected.
```

You can also display information about jobs currently running. The following sample query lists the session identifier, job number, user who submitted the job, and the start times for all currently running jobs:

```
SVRMGR> SELECT sid, r.job, log_user, r.this_date, r.this_sec
     2>    FROM dba_jobs_running r, dba_jobs j
     3>    WHERE r.job = j.job;

SID        JOB        LOG_USER             THIS_DATE THIS_SEC
---------- ---------- -------------------- --------- --------
        12      14144 JFEE                 24–OCT–94 17:21:24
        25       8536 SCOTT                24–OCT–94 16:45:12
2 rows selected.
```

See Also: For more information on data dictionary views, see the *Oracle7 Server Reference.*

PART

III

Database Storage

CHAPTER

8 Managing Tablespaces

This chapter describes the various aspects of tablespace management, and includes the following topics:

- Guidelines for Managing Tablespaces
- Creating Tablespaces
- Managing Tablespace Allocation
- Altering Tablespace Availability
- Making a Tablespace Read–Only
- Dropping Tablespaces
- Viewing Information about Tablespaces

This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide*.

Guidelines for Managing Tablespaces

Before working with tablespaces of an Oracle7 database, consider the guidelines in the following sections:

- Using Multiple Tablespaces
- Specifying Tablespace Storage Parameters
- Assigning Tablespace Quotas to Users

Using Multiple Tablespaces

Using multiple tablespaces allows you more flexibility in performing database operations. For example, when a database has multiple tablespaces, you can perform the following tasks:

- Separate user data from data dictionary data.
- Separate one application's data from another's.
- Store different tablespaces' datafiles on separate disk drives to reduce I/O contention.
- Separate rollback segment data from user data, preventing a single disk failure from causing permanent loss of data.
- Take individual tablespaces offline while others remain online.
- Reserve a tablespace for a particular type of database use, such as high update activity, read–only activity, or temporary segment storage.
- Back up individual tablespaces.

Some operating systems set a limit on the number of files that can be simultaneously open; these limits can affect the number of tablespaces that can be simultaneously online. To avoid exceeding your operating system's limit, plan your tablespaces efficiently. Create only enough tablespaces to fill your needs, and create these tablespaces with as few files as possible. If you need to increase the size of a tablespace, add one or two large datafiles, or create datafiles with the autoextend option set on, rather than many small datafiles.

Review your data in light of these advantages and decide how many tablespaces you will need for your database design.

Specifying Tablespace Storage Parameters

When you create a new tablespace, you can specify default storage parameters for objects that will be created in the tablespace. Storage parameters specified when an object is created override the default storage parameters of the tablespace containing the object. However, if you do not specify storage parameters when creating an object, the object's segment automatically uses the default storage parameters for the tablespace.

Set the default storage parameters for a tablespace to account for the size of a typical object that the tablespace will contain (you estimate this size). You can specify different storage parameters for an unusual or exceptional object when creating that object.

> **Note:** If you do not specify the default storage parameters for a new tablespace, the default storage parameters of Oracle7 become the tablespace's default storage parameters.

See Also: For information about estimating the sizes of objects, see Chapters 9 through 16.

Assigning Tablespace Quotas to Users

Grant users who will be creating tables, clusters, snapshots, indexes, and other objects, the privilege to create the object and a *quota* (space allowance or limit) in the tablespace intended to hold the object's segment. The security administrator is responsible for granting the required privileges to create objects to database users and for assigning tablespace quotas, as necessary, to database users.

See Also: To learn more about assigning tablespace quotas to database users, see page 19 – 11.

Creating Tablespaces

The steps for creating tablespaces vary by operating system. On most operating systems, you indicate the size and fully specified filenames when creating a creating a new tablespace, or altering a tablespace by adding datafiles. In each situation, Oracle7 automatically allocates and formats the datafiles as specified. However, on some operating systems, you must create the datafiles before installation.

The first tablespace in any database is always the SYSTEM tablespace. Therefore, the first datafiles of any database are automatically allocated for the SYSTEM tablespace during database creation.

You might create a new tablespace for any of the following reasons:

- You want to allocate more disk storage space for the associated database, thereby enlarging the database.
- You need to create a logical storage structure in which to store a specific type of data separate from other database data.

To increase the total size of the database you can alternatively add a datafile to an existing tablespace, rather than adding a new tablespace.

Note: No data can be inserted into any tablespace until the current instance has acquired at least two rollback segments (including the SYSTEM rollback segment).

To create a new tablespace, use either the Create Tablespace property sheet of Server Manager / GUI, or the SQL command CREATE TABLESPACE. You must have the CREATE TABLESPACE system privilege to create a tablespace.

As an example, let's create the tablespace RB_SEGS (to hold rollback segments for the database), with the following characteristics:

- The data of the new tablespace is contained in a single datafile, 50M in size.
- The default storage parameters for any segments created in this tablespace are explicitly set.
- After the tablespace is created, it is left offline.

The following statement creates the tablespace RB_SEGS:

```
CREATE TABLESPACE rb_segs
   DATAFILE 'datafilers_1' SIZE 50M
   DEFAULT STORAGE (
      INITIAL 50K
      NEXT 50K
      MINEXTENTS 2
      MAXEXTENTS 50
      PCTINCREASE 0)
   OFFLINE;
```

If you do not fully specify filenames when creating tablespaces, the corresponding datafiles are created in the current directory of the database server.

See Also: See your operating system–specific Oracle documentation for information about initially creating a tablespace.

For more information about adding a datafile, see "Adding Datafiles to a Tablespace" on page 9 – 4.

For more information about the CREATE TABLESPACE statement, see the *Oracle7 Server SQL Reference*.

Creating a Temporary Tablespace

If you wish to improve the concurrency of multiple sort operations, reduce their overhead, or avoid Oracle space management operations altogether, you can create *temporary tablespaces*.

Within a temporary tablespace, all sort operations for a given instance and tablespace share a single *sort segment*. Sort segments exist in every instance that performs sort operations within a given tablespace. You cannot store permanent objects in a temporary tablespace. You can view the allocation and deallocation of space in a temporary tablespace sort segment via the V$SORT_SEGMENTS table.

To identify a tablespace as temporary during tablespace creation, issue the following statement:

```
CREATE TABLESPACE tablespace TEMPORARY
```

To identify a tablespace as temporary in an existing tablespace, issue the following statement:

```
ALTER TABLESPACE tablespace TEMPORARY
```

> **Note:** You can take temporary tablespaces offline. Returning temporary tablespaces online does not affect their temporary status.

See Also: For more information about the CREATE TABLESPACE and ALTER TABLESPACE commands, see the *Oracle7 Server SQL Reference.*

For more information about V$SORT_SEGMENTS, see the *Oracle7 Server Reference.*

For more information about Oracle space management, see *Oracle7 Server Concepts.*

Managing Tablespace Allocation

This section describes aspects of managing tablespace allocation, and includes the following topics:

- Altering Storage Settings for Tablespaces
- Coalescing Free Space

Altering Storage Settings for Tablespaces

You can change the default storage parameters of a tablespace to change the default specifications for *future* objects created in the tablespace. To change the default storage parameters for objects subsequently created in the tablespace, use either the Alter Tablespace property sheet of Server Manager/GUI, or the SQL command ALTER TABLESPACE. Also, to alter the default storage parameters of a tablespace, you must have the ALTER TABLESPACE system privilege.

The following example alters the default storage parameters for the tablespace USERS:

```
ALTER TABLESPACE users
   DEFAULT STORAGE (
      INITIAL 50K
      NEXT 50K
      MINEXTENTS 2
      MAXEXTENTS 20
      PCTINCREASE 50);
```

New values for the default storage parameters of a tablespace affect only future extents allocated for the segments within the tablespace.

Coalescing Free Space

Space for tablespace segments is managed using extents, which are comprised of a specific number of contiguous data blocks. The free extent closest in size to the required extent is used when allocating new extents to a tablespace segment. Thus, a larger free extent can be fragmented, or smaller contiguous free extents can be coalesced into one larger free extent (see Figure 8 – 1). However, continuous allocation and deallocation of free space fragments your tablespace and makes allocation of larger extents more difficult. By default, SMON (system monitor) processes incrementally coalesce the free extents of tablespaces in the background. If desired, you can disable SMON coalescing.

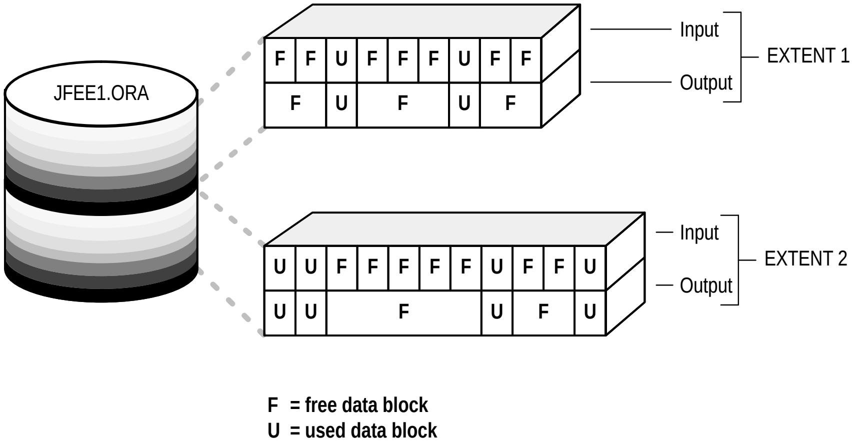


Figure 8 – 1 Coalescing Free Space

If you find that fragmentation of space is high (contiguous space on your disk appears as non–contiguous), you can coalesce your free space in a single space transaction. After every eight coalesces the space transaction commits and other transactions can allocate or deallocate space. You must have ALTER TABLESPACE privileges to coalesce tablespaces. You can coalesce all available free space extents in a tablespace into larger contiguous extents on a per tablespace basis by using the following command:

```
ALTER TABLESPACE tablespace COALESCE;
```

You can also use this command to supplement SMON and extent allocation coalescing, thereby improving space allocation performance in severely fragmented tablespaces. Issuing this command does not effect the performance of other users accessing the same tablespace. Like other options of the ALTER TABLESPACE command, the COALESCE option is exclusive; when specified, it should be the only option.

Viewing Information about Tablespaces

To display statistics about coalesceable extents for tablespaces, you can view the DBA_FREE_SPACE_COALESCED view. You can query this view to determine if you need to coalesce space in a particular tablespace.

See Also: For information about the contents of DBA_FREE_SPACE_COALESCED, see the *Oracle7 Server Reference.*

Altering Tablespace Availability

You can bring an offline tablespace online to make the schema objects within the tablespace available to database users. Alternatively, you can take an online tablespace offline while the database is open, so that this portion of the database is temporarily unavailable for general use but the rest is open and available. This section includes the following topics:

- Bringing Tablespaces Online
- Taking Tablespaces Offline

Bringing Tablespaces Online

You can bring any tablespace in an Oracle database online whenever the database is open. The only exception is that the SYSTEM tablespace must always be online because the data dictionary must always be available to Oracle. A tablespace is normally online so that the data contained within it is available to database users.

To bring an offline tablespace online while the database is open, use either the Place Online menu item of Server Manager/GUI, or the SQL command ALTER TABLESPACE. You must have the MANAGE TABLESPACE system privilege to bring a tablespace online.

> **Note:** If a tablespace to be brought online was not taken offline "cleanly" (that is, using the NORMAL option of the ALTER

TABLESPACE OFFLINE command), you must first perform media recovery on the tablespace before bringing it online. Otherwise, Oracle7 returns an error and the tablespace remains offline.

The following statement brings the USERS tablespace online:

```
ALTER TABLESPACE users ONLINE;
```

Taking Tablespaces Offline

You may wish to take a tablespace offline for any of the following reasons:

- To make a portion of the database unavailable while allowing normal access to the remainder of the database.
- To perform an offline tablespace backup (even though a tablespace can be backed up while online and in use).
- To make an application and its group of tables temporarily unavailable while updating or maintaining the application.

To take an online tablespace offline while the database is open, use either the Take Offline menu item of Server Manager / GUI, or the SQL command ALTER TABLESPACE. You must have the MANAGE TABLESPACE system privilege to take a tablespace offline.

You can specify any of the following priorities when taking a tablespace offline:

normal offline

A tablespace can be taken offline *normally* if no error conditions exist for any of the datafiles of the tablespace. No datafile in the tablespace can be currently offline as the result of a write error. With normal offline priority, Oracle7 takes a checkpoint for all datafiles of the tablespace as it takes them offline.

temporary offline

A tablespace can be taken offline *temporarily*, even if there are error conditions for one or more files of the tablespace. With temporary offline priority, Oracle7 takes offline the datafiles that are not already offline, checkpointing them as it does so.

If no files are offline, but you use the temporary option, media recovery is not required to bring the tablespace back online. However, if one or more files of the tablespace are offline because of write errors, and you take the tablespace offline temporarily, the tablespace will require recovery before you can bring it back online.

immediate offline — A tablespace can be taken offline *immediately,* without Oracle's taking a checkpoint on any of the datafiles. With immediate offline priority, media recovery for the tablespace is required before the tablespace can be brought online. You cannot take a tablespace offline immediately if the database is running in NOARCHIVELOG mode.

⚠ **Warning:** If you must take a tablespace offline, use the normal option (the default) if possible; this guarantees that the tablespace will not require recovery to come back online, even if you reset the redo log sequence (using an ALTER DATABASE OPEN RESETLOGS statement after incomplete media recovery) before bringing the tablespace back online.

Take a tablespace offline temporarily only when you cannot take it offline normally; in this case, only the files taken offline because of errors need to be recovered before the tablespace can be brought online. Take a tablespace offline immediately only after trying both the normal and temporary options.

The following example takes the USERS tablespace offline normally:

```
ALTER TABLESPACE users OFFLINE NORMAL;
```

See Also: Before taking an online tablespace offline, verify that the tablespace contains no active rollback segments. For more information see "Taking Rollback Segments Offline" on page 17 – 12.

Making a Tablespace Read–Only

This section describes issues related to making tablespaces read–only, and includes the following topics:

- Prerequisites
- Making a Read–Only Tablespace Writeable
- Creating a Read–Only Tablespace on a WORM Device

Making a tablespace read–only prevents further write operations on the datafiles in the tablespace. After making the tablespace read–only, you should back it up.

Use the SQL command ALTER TABLESPACE to change a tablespace to read–only. You must have the ALTER TABLESPACE system privilege to make a tablespace read–only. The following statement makes the FLIGHTS tablespace read–only:

```
ALTER TABLESPACE flights READ ONLY
```

After a tablespace is read–only, you can copy its files to read–only media. You must then rename the datafiles in the control file to point to the new location by using the SQL command ALTER DATABASE RENAME.

A read–only tablespace is neither online nor offline. Issuing the ALTER TABLESPACE command with the ONLINE or OFFLINE option does not change the read–only state of the tablespace; rather, it causes all of the datafiles in the tablespace to be brought online or offline.

Prerequisites

Before you can make a tablespace read–only, the following conditions must be met. It may be easiest to meet these restrictions by performing this function in restricted mode, so that only users with the RESTRICTED SESSION system privilege can be logged on.

- The tablespace must be online.
- There must not be any active transactions in the entire database.

  This is necessary to ensure that there is no undo information that needs to be applied to the tablespace.
- The tablespace must not contain any active rollback segments.

  For this reason, the SYSTEM tablespace can never be made read–only, since it contains the SYSTEM rollback segment. Additionally, because the rollback segments of a read–only tablespace are not accessible, it is recommended that you drop the rollback segments before you make a tablespace read–only.
- The tablespace must not currently be involved in an online backup, since the end of a backup updates the header file of all datafiles in the tablespace.
- The COMPATIBLE initialization parameter must be set to 7.1.0 or greater.

For better performance while accessing data in a read–only tablespace, you might want to issue a query that accesses all of the blocks of the tables in the tablespace just before making it read–only. A simple query, such as SELECT COUNT (*), executed against each table will ensure that the data blocks in the tablespace can be subsequently accessed most efficiently. This eliminates the need for Oracle7 to check the status of the transactions that most recently modified the blocks.

⚠ **Warning:** You cannot rename or resize datafiles belonging to a read–only tablespace.

See Also: For more information about read–only tablespaces, see *Oracle7 Server Concepts*.

Making a Read–Only Tablespace Writeable

Whenever you create a tablespace, it is both readable and writeable. To change a read–only tablespace back to a read–write tablespace, use the SQL command ALTER TABLESPACE. You must have the ALTER TABLESPACE system privilege to change a read–only tablespace to a read–write tablespace. The following command makes the FLIGHTS tablespace writeable:

```
ALTER TABLESPACE flights READ WRITE;
```

Making a read–only tablespace writeable updates the control file for the datafiles, so that you can use the read–only version of the datafiles as a starting point for recovery.

Prerequisites

To issue this command, all of the datafiles in the tablespace must be online. Use the DATAFILE ONLINE option of the ALTER DATABASE command to bring a datafile online. The V$DATAFILE view lists the current status of a datafile.

Creating a Read–Only Tablespace on a WORM Device

You may wish to create a read–only tablespace on a WORM (Write Once Read Many) device when you have read–only files that do not require updating.

To Create a Read–Only Tablespace on a WORM Device

1. Create a writeable tablespace on another device. Create the objects that belong in the tablespace and insert your data.
2. Issue the ALTER TABLESPACE command with the READ ONLY option to change the tablespace to read–only.
3. Copy the datafiles of the tablespace onto the WORM device. Use operating system commands to copy the files.
4. Take the tablespace offline.
5. Rename the datafiles to coincide with the names of the datafiles you copied onto your WORM device. Renaming the datafiles changes their names in the control file.
6. Bring the tablespace online.

Dropping Tablespaces

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. Any tablespace in an Oracle7 database, except the SYSTEM tablespace, can be dropped. You must have the DROP TABLESPACE system privilege to drop a tablespace.

⚠ **Warning:** Once a tablespace has been dropped, the tablespace's data is not recoverable. Therefore, make sure that all data contained in a tablespace to be dropped will not be required in the future. Also, immediately before and after dropping a tablespace from a database, back up the database completely. This is *strongly recommended* so that you can recover the database if you mistakenly drop a tablespace, or if the database experiences a problem in the future after the tablespace has been dropped.

When you drop a tablespace, only the file pointers in the control files of the associated database are dropped. The datafiles that constituted the dropped tablespace continue to exist. To free previously used disk space, delete the datafiles of the dropped tablespace using the appropriate commands of your operating system after completing this procedure.

You cannot drop a tablespace that contains any active segments. For example, if a table in the tablespace is currently being used or the tablespace contains an active rollback segment, you cannot drop the tablespace. For simplicity, take the tablespace offline before dropping it.

After a tablespace is dropped, the tablespace's entry remains in the data dictionary (see the DBA_TABLESPACES view), but the tablespace's status is changed to INVALID.

To drop a tablespace, use either the Drop tablespace menu item of Server Manager/GUI, or the SQL command DROP TABLESPACE. The following statement drops the USERS tablespace, including the segments in the tablespace:

```
DROP TABLESPACE users INCLUDING CONTENTS;
```

If the tablespace is empty (does not contain any tables, views, or other structures), you do not need to check the Including Contained Objects checkbox. If the tablespace contains any tables with primary or unique keys referenced by foreign keys of tables in other tablespaces and you want to cascade the drop of the FOREIGN KEY constraints of the child tables, select the Cascade Drop of Integrity Constraints checkbox to drop the tablespace.

Use the CASCADE CONSTRAINTS option to cascade the drop of the FOREIGN KEY constraints in the child tables.

See Also: For more information about taking tablespaces offline, see “Taking Tablespaces Offline” on page 8 – 8.

For more information about the DROP TABLESPACE statement, see the *Oracle7 Server SQL Reference*.

Viewing Information About Tablespaces

The following data dictionary views provide useful information about tablespaces of a database:

- USER_EXTENTS, DBA_EXTENTS
- USER_SEGMENTS, DBA_SEGMENTS
- USER_FREE_SPACE, DBA_FREE_SPACE
- DBA_USERS
- DBA_TS_QUOTAS
- USER_TABLESPACES, DBA_TABLESPACES
- DBA_DATA_FILES
- V$DATAFILE

The following examples illustrate how to use the views not already illustrated in other chapters of this manual. They assume you are using a database that contains two tablespaces, SYSTEM and USERS. USERS is made up of two files, FILE1 (100MB) and FILE2 (200MB); the tablespace has been taken offline normally.

Listing Tablespaces and Default Storage Parameters: Example

To list the names and default storage parameters of all tablespaces in a database, use the following query on the DBA_TABLESPACES view:

```
SELECT tablespace_name ”TABLESPACE”,
   initial_extent ”INITIAL_EXT”,
   next_extent ”NEXT_EXT”,
   min_extents ”MIN_EXT”,
   max_extents ”MAX_EXT”,
   pct_increase
   FROM sys.dba_tablespaces;

TABLESPACE INITIAL_EXT NEXT_EXT MIN_EXT MAX_EXT PCT_INCREASE
---------- ----------- -------- ------- ------- ------------
SYSTEM        10240000 10240000       1      99           50
```

```
USERS         10240000 10240000       1      99           50
```

Listing the Datafiles and Associated Tablespaces of a Database: Example

To list the names, sizes, and associated tablespaces of a database, enter the following query on the DBA_DATA_FILES view:

```
SELECT  file_name, bytes, tablespace_name
   FROM sys.dba_data_files;

FILE_NAME    BYTES      TABLESPACE_NAME
------------ ---------- --------------------
filename1    10240000   SYSTEM
filename2    10240000   USERS
filename3    20480000   USERS
```

Listing the Free Space (Extents) of Each Tablespace: Example

To see the amount of space available in the free extents of each tablespace in the database, enter the following query:

```
SELECT tablespace_name, file_id,
   COUNT(*)    ”PIECES”,
   MAX(blocks) ”MAXIMUM”,
   MIN(blocks) ”MINIMUM”,
   AVG(blocks) ”AVERAGE”,
   SUM(blocks) ”TOTAL”
   FROM sys.dba_free_space
WHERE tablespace_name = ’SYSTEM’
GROUP BY tablespace_name, file_id;


TABLESPACE FILE_ID PIECES MAXIMUM MINIMUM AVERAGE     SUM
---------- ------- ------ ------- ------- ------- -------
SYSTEM           1      2    2928     115  1521.5    3043
```

SUM shows the amount of free space in each tablespace, PIECES shows the amount of fragmentation in the datafiles of the tablespace, and MAXIMUM shows the largest contiguous area of space. This query is useful when you are going to create a new object or you know that a segment is about to extend, and you want to make sure that there is enough space in the containing tablespace.

CHAPTER

9 Managing Datafiles

This chapter describes the various aspects of datafile management, and includes the following topics:

- Guidelines for Managing Datafiles
- Creating and Adding Datafiles to a Tablespace
- Changing a Datafile's Size
- Altering Datafile Availability
- Renaming and Relocating Datafiles
- Verifying Data Blocks in Datafiles
- Viewing Information About Datafiles

See Also: This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide.*

Datafiles can also be created as part of database recovery from a media failure. For more information, see page 24 – 7.

For information on tablespaces and datafiles in Trusted Oracle7 Server, see the *Trusted Oracle7 Server Administrator's Guide.*

Guidelines for Managing Datafiles

This section describes aspects of managing datafiles, and includes the following topics:

- Number of Datafiles
- Set the Size of Datafiles
- Place Datafiles Appropriately
- Store Datafiles Separately From Redo Log FIles

Number of Datafiles

At least one datafile is required for the SYSTEM tablespace of a database; a small system might have a single datafile. In general, keeping a few large datafiles is preferable to many small datafiles, because you can keep fewer files open at the same time.

You can add datafiles to tablespaces, subject to the following operating system–specific datafile limits:

operating system limit

Each operating system sets a limit on the maximum number of files per process. Regardless of all other limits, more datafiles cannot be created when the operating system limit of open files is reached.

Oracle7 system limit

Oracle7 imposes a maximum limit on the number of datafiles for any Oracle7 database opened by any instance. This limit is port–specific.

control file upper bound

At database creation, you must indicate the maximum number of datafiles expected for the database so that an adequate amount of space can be reserved in the database's control file. You set this limit with the MAXDATAFILES parameter in the CREATE DATABASE statement. This maximum cannot exceed the Oracle7 system limit or any operating system limit. If you are not sure how to set this parameter, use a high number to avoid unnecessary limitation. The default value is operating system–specific.

Note: You can increase space in the database by resizing datafiles. Resizing existing datafiles is useful if you are nearing the MAXDATAFILES limit.

instance or SGA upper bound

When starting an Oracle7 instance, the database's parameter file indicates the amount of SGA space to reserve for datafile information; the maximum number of datafiles is controlled by the DB_FILES parameter. This limit applies only for the life of the instance. DB_FILES can temporarily reduce the control file upper bound, but cannot raise it. The default value is the value of the control file upper bound.

With the Oracle7 Parallel Server, all instances must set the instance datafile upper bound to the same value.

The use of DB_FILES and MAXDATAFILES is optional. If neither is used, the default maximum number of datafiles is the operating system–specific Oracle7 system limit.

See Also: For more information on operating system limits, see your operating system–specific Oracle documentation.

For information about Parallel Server operating system limits, see the *Oracle7 Parallel Server* manual.

For more information about MAXDATAFILES, see the *Oracle7 Server SQL Reference.*

Set the Size of Datafiles

The first datafile (in the original SYSTEM tablespace) must be at least 2M to contain the initial data dictionary and rollback segment. If you install other Oracle7 products, they may require additional space in the SYSTEM tablespace (for online help, for example); see the installation instructions for these products.

Place Datafiles Appropriately

Tablespace location is determined by the physical location of the datafiles that constitute that tablespace. Use the hardware resources of your computer appropriately.

For example, if several disk drives are available to store the database, it might be helpful to store table data in a tablespace on one disk drive, and index data in a tablespace on another disk drive. This way, when users query table information, both disk drives can work simultaneously, retrieving table and index data at the same time.

Store Datafiles Separately From Redo Log Files

Datafiles should not be stored on the same disk drive that stores the database's redo log files. If the datafiles and redo log files are stored on the same disk drive and that disk drive fails, the files cannot be used in your database recovery procedures.

If you multiplex your redo log files, then the likelihood of your losing all of your redo log files is low, so you can store datafiles on the same drive as some redo log files.

Creating and Adding Datafiles to a Tablespace

You can create and add datafiles to a tablespace to increase the total amount of disk space allocated for the tablespace, and consequently the database.

To add datafiles to a tablespace, use either the Add Datafile dialog box of Server Manager/GUI, or the SQL command ALTER TABLESPACE. You must have the ALTER TABLESPACE system privilege to add datafiles to a tablespace.

The following statement creates a new datafile for the RB_SEGS tablespace:

```
ALTER TABLESPACE rb_segs
   ADD DATAFILE 'filename1' SIZE 1M;
```

If you add new datafiles to a tablespace and do not fully specify the filenames, Oracle7 creates the datafiles in the default directory of the database server. Unless you want to reuse existing files, make sure the new filenames do not conflict with other files; the old files that have been previously dropped will be overwritten.

Changing a Datafile's Size

This section describes the various ways to alter the size of a datafile, and includes the following topics:

- Enabling and Disabling Automatic Extension for a Datafile
- Manually Resizing a Datafile

Enabling and Disabling Automatic Extension for a Datafile

You can create datafiles or alter existing datafiles so that they automatically increase in size when more space is needed in the database. The files increase in specified increments up to a specified maximum.

Setting your datafiles to extend automatically results in the following:

- reduces the need for immediate intervention when a tablespace runs out of space
- ensures applications will not halt because of failures to allocate extents
- creates FILEXT$, which contains information about the autoextend characteristics of a datafile

> **Note:** FILEXT$ is not created with the data dictionary scripts, so there is no easy way to create database views on it. Thus, catalog scripts will be unsuccessful if you attempt to create a database view on FILEXT$. FILEXT$ is the only place you can query where datafiles have autoextend turned on, and the current settings of their parameter values.

You can specify automatic file extension when you create datafiles via the following SQL commands:

- CREATE DATABASE
- CREATE TABLESPACE
- ALTER TABLESPACE

You can enable or disable automatic file extension for existing datafiles, or manually resize a datafile using the SQL command ALTER DATABASE.

The following example enables automatic extension for a datafile, FILENAME2, added to the USERS tablespace:

```
ALTER TABLESPACE users
   ADD DATAFILE ’filename2’ SIZE 10M
      AUTOEXTEND ON
      NEXT 512K
      MAXSIZE 250M
```

The value of NEXT is the minimum size of the increments added to the file when it extends. The value of MAXSIZE is the maximum size to which the file can automatically extend.

The next example disables automatic extension for the datafile FILENAME2:

```
ALTER DATABASE DATAFILE ’filename2’
   AUTOEXTEND OFF
```

See Also: For more information about the SQL commands for creating or altering datafiles, see the *Oracle7 Server SQL Reference*.

Manually Resizing a Datafile

You can manually increase or decrease the size of a datafile using the ALTER DATABASE command.

Because you can change the sizes of datafiles, you can add more space to your database without adding more datafiles. This is beneficial if you are concerned about reaching the maximum number of datafiles allowed in your database.

Manually reducing the sizes of datafiles allows you to reclaim unused space in the database. This is useful for correcting errors in estimates of space requirements.

In this example, assume that the datafile FILENAME2 has extended up to 250M. However, because its tablespace now stores smaller objects, the datafile can be reduced in size.

The following command decreases the size of datafile FILENAME2:

```
ALTER DATABASE DATAFILE 'filename2'
   RESIZE 100M
```

> **Note:** It is not always possible to decrease the size of a file to a specific value.

See Also: For more information about the implications resizing files has for downgrading, see the *Oracle7 Server Migration*.

For more information about the ALTER DATABASE command, see the *Oracle7 Server SQL Reference*.

Altering Datafile Availability

This section describes ways to alter datafile availability, and includes the following topics:

- Bringing Datafiles Online in ARCHIVELOG Mode
- Taking Datafiles Offline in NOARCHIVELOG Mode

In very rare situations, you might need to bring specific datafiles online (make them available) or take specific files offline (make them unavailable). For example, when Oracle7 has problems writing to a datafile, it can automatically take the datafile offline. You might need to take the damaged datafile offline or bring it online manually.

> **Note:** You can make all datafiles in a tablespace, other than the files in the SYSTEM tablespace, temporarily unavailable by taking the tablespace offline. You *must* leave these files in the tablespace to bring the tablespace back online.

Offline datafiles cannot be accessed. Bringing a datafile in a read–only tablespace online makes the file readable. No one can write to the file unless its associated tablespace is returned to the read–write state. The files of a read–only tablespace can independently be taken online or offline using the DATAFILE option of the ALTER DATABASE command.

To bring a datafile online or take it offline, in either archiving mode, you must have the ALTER DATABASE system privilege. You can perform these operations only when the database is open in exclusive mode.

Bringing Datafiles Online in ARCHIVELOG Mode

To bring an individual datafile online, issue the SQL command ALTER DATABASE, and include the DATAFILE parameter.

> **Note:** To use this option of the ALTER DATABASE command, the database must be in ARCHIVELOG mode. This requirement prevents you from accidentally losing the datafile, since taking the datafile offline while in NOARCHIVELOG mode is likely to result in losing the file.

The following statement brings the specified datafile online:

```
ALTER DATABASE DATAFILE 'filename' ONLINE;
```

See Also: For more information about bringing datafiles online during media recovery, see page 24 – 7.

Taking Datafiles Offline in NOARCHIVELOG Mode

To take a datafile offline when the database is in NOARCHIVELOG mode, use the ALTER DATABASE command with the DATAFILE parameter and the OFFLINE DROP option. This allows you to take the datafile offline and drop it immediately. It is useful, for example, if the datafile contains only data from temporary segments and has not been backed up, and the database is in NOARCHIVELOG mode.

The following statement brings the specified datafile offline:

```
ALTER DATABASE DATAFILE 'filename' OFFLINE DROP;
```

Renaming and Relocating Datafiles

This section describes the various aspects of renaming and relocating datafiles, and includes the following topics:

- Renaming and Relocating Datafiles for a Single Tablespace
- Renaming and Relocating Datafiles for Multiple Tablespaces

You can rename datafiles to change either their names or locations. Oracle7 provides options to make the following changes:

- Rename and relocate datafiles in a single offline tablespace (for example, FILENAME1 and FILENAME2 in TBSPACE1) while the rest of the database is open.

- Rename and relocate datafiles in several tablespaces simultaneously (for example, FILE1 in TBSP1 and FILE2 in TBSP2) while the database is mounted but closed.

  Note: To rename or relocate datafiles of the SYSTEM tablespace, you must use the second option, because you cannot take the SYSTEM tablespace offline.

Renaming and relocating datafiles with these procedures only change the pointers to the datafiles, as recorded in the database's control file; it does not physically rename any operating system files, nor does it copy files at the operating system level. Therefore, renaming and relocating datafiles involve several steps. Read the steps and examples carefully before performing these procedures.

You must have the ALTER TABLESPACE system privilege to rename datafiles of a single tablespace.

Renaming and Relocating Datafiles for a Single Tablespace

The following steps describe how to rename or relocate datafiles from a single tablespace.

To Rename or Relocate Datafiles for a Single Tablespace

1. Take the non–SYSTEM tablespace that contains the datafiles offline.
2. Copy the datafiles to the new location or new names using the operating system.
3. Make sure that the new, fully specified filenames are different from the old filenames.
4. Use either the Rename Datafile dialog box of Server Manager/GUI or the SQL command ALTER TABLESPACE with the RENAME DATAFILE option to change the filenames within the database.

For example, the following statement renames the datafiles FILENAME1 and FILENAME2 to FILENAME3 and FILENAME4, respectively:

```
ALTER TABLESPACE users
   RENAME DATAFILE 'filename1', 'filename2'
      TO 'filename3', 'filename4';
```

The new file must already exist; this command does not create a file. Also, always provide complete filenames (including their paths) to properly identify the old and new datafiles. In particular, specify the old filename exactly as it appears in the DBA_DATA_FILE view of the data dictionary.

Renaming and Relocating Datafiles for Multiple Tablespaces

You can rename and relocate datafiles of one or more tablespaces with the SQL command ALTER DATABASE with the RENAME FILE option. This option is the only choice if you want to rename or relocate datafiles of several tablespaces in one operation, or rename or relocate datafiles of the SYSTEM tablespace. If the database must remain open, consider instead the procedure outlined in the previous section.

To rename datafiles of several tablespaces in one operation or to rename datafiles of the SYSTEM tablespace, you must have the ALTER DATABASE system privilege.

To Rename and Relocate Datafiles for Multiple Tablespaces

1. Ensure that the database is mounted but closed.
2. Copy the datafiles to be renamed to their new locations and new names, using the operating system.
3. Make sure the new copies of the datafiles have different fully specified filenames from the datafiles currently in use.
4. Use the SQL command ALTER DATABASE to rename the file pointers in the database's control file.

For example, the following statement renames the datafiles FILENAME 1 and FILENAME2 to FILENAME3 and FILENAME4, respectively:

```
ALTER DATABASE
   RENAME FILE ’filename1’, ’filename2’
   TO ’filename3’, ’filename4’;
```

The new file must already exist; this command does not create a file. Also, always provide complete filenames (including their paths) to properly identify the old and new datafiles. In particular, specify the old filename exactly as it appears in the DBA_DATA_FILE view of the data dictionary.

Relocating Datafiles: Example

For this example, assume the following conditions:

- An open database has a tablespace named USERS that is comprised of datafiles located on the same disk of a computer.
- The datafiles of the USERS tablespace are to be relocated to a different disk drive.
- You are currently connected with administrator privileges to the open database while using Server Manager.

To Relocate Datafiles

1. Identify the datafile names of interest.

   The following query of the data dictionary view DBA_DATA_FILES lists the datafile names and respective sizes (in bytes) of the USERS tablespace:

   ```
   SELECT file_name, bytes FROM sys.dba_data_files
      WHERE tablespace_name = ’USERS’;
   FILE_NAME         BYTES
   ---------------------------
   FILENAME1         102400000
   FILENAME2         102400000
   ```

   Here, FILENAME1 and FILENAME2 are two fully specified filenames, each 1MB in size.

2. Back up the database.

   Before making any structural changes to a database, such as renaming and relocating the datafiles of one or more tablespaces, always completely back up the database.

3. Take the tablespace containing the datafile offline, or shut down the database and restart and mount it, leaving it closed. Either option closes the datafiles of the tablespace.

4. Copy the datafiles to their new locations using operating system commands. For this example, the existing files FILENAME1 and FILENAME2 are copied to FILENAME3 and FILENAME4.

   Suggestion: You can execute an operating system command to copy a file without exiting Server Manager/LineMode by using the HOST command.

5. Rename the datafiles within Oracle.

   The datafile pointers for the files that comprise the USERS tablespace, recorded in the control file of the associated database, must now be changed from FILENAME1 and FILENAME2 to FILENAME3 and FILENAME4, respectively.

   If the tablespace is offline but the database is open, use the Server Manager Rename Datafiles dialog box or ALTER TABLESPACE...RENAME DATAFILE command. If the database is mounted but closed, use the ALTER DATABASE...RENAME FILE command.

6. Bring the tablespace online, or shut down and restart the database.

   If the USERS tablespace is offline and the database is open, bring the tablespace back online. If the database is mounted but closed, open the database.

7. Back up the database. After making any structural changes to a database, always perform an immediate and complete backup.

See Also: For more information about the DBA_DATA_FILES data dictionary view, see the *Oracle7 Server Reference*.

For more information about taking a tablespace offline, see "Taking Tablespaces Offline" on page 8 – 8.

For more information about mounting a database without opening it, see Chapter 3.

Verifying Data Blocks in Datafiles

If you want to configure Oracle7 to use checksums to verify data blocks, set the initialization parameter DB_BLOCK_CHECKSUM to TRUE. The default value of DB_BLOCK_CHECKSUM is FALSE.

When you enable block checking, Oracle7 computes a checksum for each block written to disk. Checksums are computed for all data blocks, including temporary blocks.

The DBWR process calculates the checksum for each block and stores it in the block's header. Checksums are also computed by the direct loader.

The next time Oracle7 reads a data block, it uses the checksum to detect corruption in the block. If a corruption is detected, Oracle7 returns message ORA–01578 and writes information about the corruption to a trace file.

⚠ **Warning:** Setting DB_BLOCK_CHECKSUM to TRUE can cause performance overhead. Set this parameter to TRUE only under the advice of Oracle Support personnel to diagnose data corruption problems.

Viewing Information About Datafiles

The following data dictionary views provide useful information about the datafiles of a database:

- USER_EXTENTS, DBA_EXTENTS
- USER_SEGMENTS, DBA_SEGMENTS
- USER_FREE_SPACE, DBA_FREE_SPACE
- DBA_USERS
- DBA_TS_QUOTAS
- USER_TABLESPACES, DBA_TABLESPACES
- DBA_DATA_FILES
- V$DATAFILE

Listing Status Information About Datafiles: Example

The following example illustrates how to use a view not already illustrated in other chapters of this manual. Assume you are using a database that contains two tablespaces, SYSTEM and USERS. USERS is made up of two files, FILE1 (100MB) and FILE2 (200MB); the tablespace has been taken offline normally. Here, you query V$DATAFILE to view status information about datafiles of a database:

```
SELECT name,
   file#,
   status,
   checkpoint_change# ”CHECKPOINT”   FROM v$datafile;

NAME                                FILE# STATUS   CHECKPOINT
--------------------------------- ----- ------- ----------
filename1                           1     SYSTEM        3839
filename2                           2     OFFLINE       3782
filename3                           3     OFFLINE       3782
```

FILE# lists the file number of each datafile; the first datafile in the SYSTEM tablespace, created with the database, is always file 1. STATUS lists other information about a datafile. If a datafile is part of the SYSTEM tablespace, its status is SYSTEM (unless it requires recovery). If a datafile in a non–SYSTEM tablespace is online, its status is ONLINE. If a datafile in a non–SYSTEM tablespace is offline, its status can be either OFFLINE or RECOVER. CHECKPOINT lists the final SCN written for a datafile’s most recent checkpoint.

CHAPTER

10

Guidelines for Managing Schema Objects

This chapter describes guidelines for managing schema objects, and includes the following topics:

- Managing Space in Data Blocks
- Setting Storage Parameters
- Deallocating Space
- Understanding Space Use of Datatypes

You should familiarize yourself with the concepts in this chapter before attempting to manage specific schema objects, as described in chapters 11 – 16.

See Also: This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide*.

Managing Space in Data Blocks

This section describes the various aspects of managing space in data blocks, and includes the following topics:

- The PCTFREE Parameter
- The PCTUSED Parameter
- Selecting Associated PCTUSED and PCTFREE Values

You can use the PCTFREE and PCTUSED parameters to make the following changes:

- increase the performance of writing and retrieving data
- decrease the amount of unused space in data blocks
- decrease the amount of row chaining between data blocks

The PCTFREE Parameter

The PCTFREE parameter is used to set the percentage of a block to be reserved for possible updates to rows that already are contained in that block. For example, assume that you specify the following parameter within a CREATE TABLE statement:

```
PCTFREE 20
```

This indicates that 20% of each data block used for this table's data segment will be kept free and available for possible updates to the existing rows already within each block. Figure 10 – 1 illustrates PCTFREE.

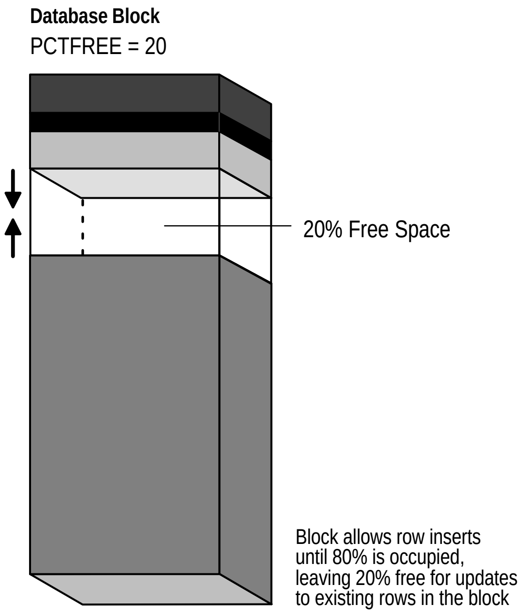

Database Block
PCTFREE = 20
20% Free Space
Block allows row inserts until 80% is occupied, leaving 20% free for updates to existing rows in the block

Figure 10 – 1 PCTFREE

Notice that before the block reaches PCTFREE, the free space of the data block is filled by both the insertion of new rows and by the growth of the data block header.

Specifying PCTFREE

The default for PCTFREE is 10 percent. You can use any integer between 0 and 99, inclusive, as long as the sum of PCTFREE and PCTUSED does not exceed 100.

A smaller PCTFREE has the following effects:

- reserves less room for updates to expand existing table rows
- allows inserts to fill the block more completely
- may save space, because the total data for a table or index is stored in fewer blocks (more rows or entries per block)

A small PCTFREE might be suitable, for example, for a segment that is rarely changed.

A larger PCTFREE has the following effects:

- reserves more room for future updates to existing table rows
- may require more blocks for the same amount of inserted data (inserting fewer rows per block)
- may improve update performance, because Oracle does not need to chain row pieces as frequently, if ever

A large PCTFREE is suitable, for example, for segments that are frequently updated.

Ensure that you understand the nature of the table or index data before setting PCTFREE. Updates can cause rows to grow. New values might not be the same size as values they replace. If there are many updates in which data values get larger, PCTFREE should be increased. If updates to rows do not affect the total row width, PCTFREE can be low. Your goal is to find a satisfactory tradeoff between densely packed data and good update performance.

PCTFREE for Non–Clustered Tables If the data in the rows of a non–clustered table is likely to increase in size over time, reserve some space for these updates. Otherwise, updated rows are likely to be chained among blocks.

PCTFREE for Clustered Tables The discussion for non–clustered tables also applies to clustered tables. However, if PCTFREE is reached, new rows from *any* table contained in the same cluster key go into a new data block that is chained to the existing cluster key.

PCTFREE for Indexes You can specify PCTFREE only when initially creating an index.

The PCTUSED Parameter

After a data block becomes full, as determined by PCTFREE, Oracle does not consider the block is for the insertion of new rows until the percentage of the block being used falls below the parameter PCTUSED. Before this value is achieved, Oracle uses the free space of the data block only for updates to rows already contained in the data block. For example, assume that you specify the following parameter within a CREATE TABLE statement:

```
PCTUSED 40
```

In this case, a data block used for this table's data segment is not considered for the insertion of any new rows until the amount of used space in the block falls to 39% or less (assuming that the block's used space has previously reached PCTFREE). Figure 10 – 2 illustrates this.

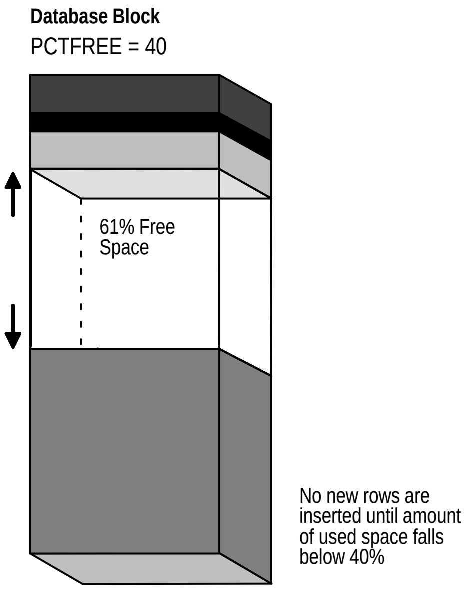

Database Block
PCTFREE = 40
61% Free Space
No new rows are inserted until amount of used space falls below 40%

Figure 10 – 2 PCTUSED

Specifying PCTUSED

Once the free space in a data block reaches PCTFREE, no new rows are inserted in that block until the percentage of space used falls below PCTUSED. The percent value is for the block space available for data after overhead is subtracted from total space.

The default for PCTUSED is 40 percent. You may specify any integer between 0 and 99, inclusive, as long as the sum of PCTUSED and PCTFREE does not exceed 100.

A smaller PCTUSED has the following effects:

- reduces processing costs incurred during UPDATE and DELETE statements for moving a block to the free list when it has fallen below that percentage of usage
- increases the unused space in a database

A larger PCTUSED has the following effects:

- improves space efficiency
- increases processing cost during INSERTs and UPDATEs

Selecting Associated PCTUSED and PCTFREE Values

If you decide not to use the default values for PCTFREE or PCTUSED, keep the following guidelines in mind:

- The sum of PCTFREE and PCTUSED must be equal to or less than 100.
- If the sum equals 100, then Oracle attempts to keep no more than PCTFREE free space, and processing costs are highest.
- Block overhead is not included in the computation of PCTUSED or PCTFREE.
- The smaller the difference between 100 and the sum of PCTFREE and PCTUSED (as in PCTUSED of 75, PCTFREE of 20), the more efficient space usage is, at some performance cost.

Examples of Choosing PCTFREE and PCTUSED Values

The following examples show how and why specific values for PCTFREE and PCTUSED are specified for tables.

Example 1

Scenario: Common activity includes UPDATE statements that increase the size of the rows.

Settings: PCTFREE = 20
PCTUSED = 40

Explanation: PCTFREE is set to 20 to allow enough room for rows that increase in size as a result of updates. PCTUSED is set to 40 so that less processing is done during high update activity, thus improving performance.

Example 2

Scenario: Most activity includes INSERT and DELETE statements, and UPDATE statements that do not increase the size of affected rows.

Settings: PCTFREE = 5
PCTUSED = 60

Explanation: PCTFREE is set to 5 because most UPDATE statements do not increase row sizes. PCTUSED is set to 60 so that space freed by DELETE statements is used soon, yet processing is minimized.

Example 3

Scenario: The table is very large; therefore, storage is a primary concern. Most activity includes read–only transactions.

Settings: PCTFREE = 5
PCTUSED = 90

Explanation: PCTFREE is set to 5 because this is a large table and you want to completely fill each block.

Setting Storage Parameters

This section describes the storage parameters you can set for various data structures, and includes the following topics:

- Storage Parameters You Can Specify
- Setting INITRANS and MAXTRANS
- Setting Default Storage Parameters for Segments in a Tablespace
- Setting Storage Parameters for Data Segments
- Setting Storage Parameters for Index Segments
- Changing Values for Storage Parameters
- Understanding Precedence in Storage Parameters

You can set storage parameters for the following types of logical storage structures:

- tablespaces (defaults for any segment in the tablespace)
- tables, clusters, snapshots, and snapshot logs (data segments)
- indexes (index segments)
- rollback segments

Storage Parameters You Can Specify

Every database has default values for storage parameters. You can specify defaults for a tablespace, which override the system defaults to become the defaults for objects created in that tablespace only. Furthermore, you can specify storage settings for each individual object. The storage parameters you can set are listed below, along with their system defaults.

INITIAL

The size, in bytes, of the first extent allocated when a segment is created.

Default: 5 data blocks
Minimum: 2 data blocks (rounded up)
Maximum: operating system–specific

Although the default system value is given in data blocks, use bytes to set a value for this parameter. You can use the abbreviations K and M to indicate kilobytes and megabytes. Anything less than 2 data blocks is rounded *up* to the next multiple of the data block size, as determined by the parameter DB_BLOCK_SIZE.

For example, if the data block size of a database is 2048 bytes, then the system default for the INITIAL storage parameter of tablespaces is 10240 bytes. If you create a tablespace in this database and specify its default storage parameter INITIAL as 20000 (bytes), Oracle automatically rounds this value up to 20480 (10 data blocks).

NEXT

The size, in bytes, of the next incremental extent to be allocated for a segment. The second extent is equal to the original setting for NEXT. From there forward, NEXT is set to the previous size of NEXT multiplied by (1 + PCTINCREASE / 100).

Default: 5 data blocks
Minimum: 1 data block
Maximum: operating system–specific

As with INITIAL, although the default system value is given in data blocks, use bytes to set a value for this parameter. You can use the abbreviations K and M to indicate kilobytes and megabytes. The value is rounded *up* to the next multiple of the data block size, as determined by the parameter DB_BLOCK_SIZE.

MAXEXTENTS

The total number of extents, including the first, that can ever be allocated for the segment.

Default: dependent on the data block size and operating system
Minimum: 1 (extent)
Maximum: unlimited

MINEXTENTS

The total number of extents to be allocated when the segment is created. This allows for a large allocation of space at creation time, even if contiguous space is not available.

Default: 1 (extent)
Minimum: 1 (extent)
Maximum: operating system–specific

If MINEXTENTS is greater than 1, then the specified number of incremental extents are allocated at creation time using the values INITIAL, NEXT, and PCTINCREASE.

Note: The default and minimum values of MINEXTENTS for a rollback segment are always 2. If you want to guarantee that you have enough space to load all the data for one table, create the table with a large MINEXTENTS value so that the LOAD operation is successful even if your database is fragmented.

PCTINCREASE

The percent by which each incremental extent grows over the last incremental extent allocated for a segment. If PCTINCREASE is 0, then all incremental extents are the same size. If PCTINCREASE is greater than zero, then each time NEXT is calculated, it grows by PCTINCREASE. PCTINCREASE cannot be negative.

The new NEXT equals 1 + PCTINCREASE / 100, multiplied by the size of the last incremental extent (the old NEXT) and rounded *up* to the next multiple of a block size.

Default: 50 (%)
Minimum: 0 (%)
Maximum: operating system–specific

Note: PCTINCREASE is always 0 for rollback segments. PCTINCREASE cannot be specified for rollback segments.

By using PCTINCREASE correctly, you can reduce the fragmentation of a segment by enlarging incremental extents and reducing the number of extents that need to be allocated for the segment. The segment contains a few large extents, rather than many smaller extents.

If you change PCTINCREASE for a segment, the current value of NEXT for that segment does not change. Only future values of NEXT are affected.

INITRANS

Reserves a pre–allocated amount of space for an initial number of transaction entries to access rows in the data block concurrently. Space is reserved in the headers of all data blocks in the associated data or index segment. The default value is 1 for tables and 2 for clusters and indexes.

MAXTRANS

As multiple transactions concurrently access the rows of the same data block, space is allocated for each transaction's entry in the block. Once the space reserved by INITRANS is depleted, space for additional transaction entries is allocated out of the free space in a block, if available. Once allocated, this space effectively becomes a permanent part of the block header. The MAXTRANS parameter limits the number of transaction entries that can concurrently use data in a data block. Therefore, you can limit the amount of free space that can be allocated for transaction entries in a data block using MAXTRANS. The default value is an operating system–specific function of block size, not exceeding 255.

If MAXTRANS is too low, transactions blocked by this limit must wait until other transactions complete and free transaction entry space. For example, if MAXTRANS is 3 and a fourth concurrent transaction attempts to access a block already being accessed by three active transactions, Oracle selects one of the three and waits until it commits or rolls back, and then proceeds with the fourth transaction.

See Also: Some defaults are operating system specific; see your operating system–specific Oracle documentation.

Setting INITRANS and MAXTRANS

Transaction entry settings for the data blocks allocated for a table, cluster, or index should be set individually for each object based on the following criteria:

- the space you would like to reserve for transaction entries compared to the space you would reserve for database data
- the number of concurrent transactions that are likely to touch the same data blocks at any given time

For example, if a table is very large and only a small number of users simultaneously access the table, the chances of multiple concurrent transactions requiring access to the same data block is low. Therefore, INITRANS can be set low, especially if space is at a premium in the database.

Alternatively, assume that a table is usually accessed by many users at the same time. In this case, you might consider pre–allocating transaction entry space by using a high INITRANS (to eliminate the overhead of having to allocate transaction entry space, as required when the object is in use) and allowing a higher MAXTRANS so that no user has to wait to access any necessary data blocks.

Setting Default Storage Parameters for Segments in a Tablespace

You can set default storage parameters for each tablespace of a database. Any storage parameter that you do not explicitly set when creating or subsequently altering a segment in a tablespace automatically is set to the corresponding default storage parameter for the tablespace in which the segment resides.

Setting Storage Parameters for Data Segments

You can set the storage parameters for the data segment of a non–clustered table, snapshot, or snapshot log using the STORAGE clause of the CREATE or ALTER statement for tables, snapshots, or snapshot logs.

In contrast, you set the storage parameters for the data segments of a cluster using the STORAGE clause of the CREATE CLUSTER or ALTER CLUSTER command, rather than the individual CREATE or ALTER commands that put tables and snapshots into the cluster. Storage parameters specified when creating or altering a *clustered* table or snapshot are ignored. The storage parameters set for the cluster override the table's storage parameters.

Setting Storage Parameters for Index Segments

Storage parameters for an index segment created for a table index can be set using the STORAGE clause of the CREATE INDEX or ALTER INDEX command. Storage parameters of an index segment created for the index used to enforce a primary key or unique key constraint can be set in the ENABLE clause of the CREATE TABLE or ALTER TABLE commands or the STORAGE clause of the ALTER INDEX command.

A PCTFREE setting for an index only has an effect when the index is created. You cannot specify PCTUSED for an index segment.

Changing Values for Storage Parameters

You can alter default storage parameters for tablespaces and specific storage parameters for individual segments if the current settings are incorrect. All default storage parameters can be reset for a tablespace. However, changes affect only new objects created in the tablespace, or new extents allocated for a segment.

The INITIAL and MINEXTENTS storage parameters cannot be altered for an existing table, cluster, index, or rollback segment. If only NEXT is altered for a segment, the next incremental extent is the size of the new NEXT, and subsequent extents can grow by PCTINCREASE as usual.

If both NEXT and PCTINCREASE are altered for a segment, the next extent is the new value of NEXT, and from that point forward, NEXT is calculated using PCTINCREASE as usual.

Understanding Precedence in Storage Parameters

The storage parameters in effect at a given time are determined by the following types of SQL statements, listed in order of precedence:

1. ALTER TABLE / CLUSTER / SNAPSHOT / SNAPSHOT LOG / INDEX / ROLLBACK SEGMENT statement
2. CREATE TABLE / CLUSTER / SNAPSHOT / SNAPSHOT LOG / CREATE INDEX / ROLLBACK SEGMENT statement
3. ALTER TABLESPACE statement
4. CREATE TABLESPACE statement
5. Oracle default statement

Any storage parameter specified at the object level overrides the corresponding option set at the tablespace level. When storage parameters are not explicitly set at the object level, they default to those at the tablespace level. When storage parameters are not set at the tablespace level, Oracle system defaults apply. If storage parameters are altered, the new options apply only to the extents not yet allocated.

> **Note:** The storage parameters for temporary segments always use the default storage parameters set for the associated tablespace.

Storage Parameter Example

Assume the following statement has been executed:

```
CREATE TABLE test_storage
   ( . . . )
   STORAGE (INITIAL 100K   NEXT 100K
      MINEXTENTS 2   MAXEXTENTS 5
      PCTINCREASE 50);
```

Also assume that the initialization parameter DB_BLOCK_SIZE is set to 2K. The following table shows how extents are allocated for the TEST_STORAGE table. Also shown is the value for the incremental extent, as can be seen in the NEXT column of the USER_SEGMENTS or DBA_SEGMENTS data dictionary views:

Extent#	Extent Size	Value for NEXT
1	100K or 50 blocks	100K
2	100K or 50 blocks	CEIL(100K*1.5)=150K
3	150K or 75 blocks	CEIL(150K*1.5)=228K
4	228K or 114 blocks	CEIL(228K*1.5)=342K
5	342K or 171 blocks	CEIL(342K*1.5)=516K

Table 10 – 1 Extent Allocations

If you change the NEXT or PCTINCREASE storage parameters with an ALTER statement (such as ALTER TABLE), the specified value replaces the current value stored in the data dictionary. For example, the following statement modifies the NEXT storage parameter of the TEST_STORAGE table before the third extent is allocated for the table:

```
ALTER TABLE test_storage STORAGE (NEXT 500K);
```

As a result, the third extent is 500K when allocated, the fourth is (500K*1.5)=750K, and so on.

Deallocating Space

This section describes aspects of deallocating unused space, and includes the following topics:

- Viewing the High Water Mark
- Issuing Space Deallocation Statements

It is not uncommon to allocate space to a segment, only to find out later that it is not being used. For example, you may set PCTINCREASE to a high value, which could create a large extent that is only partially used. Or you could explicitly overallocate space by issuing the ALTER TABLE ALLOCATE EXTENT statement. If you find that you have unused or overallocated space, you can release it so that the unused space can be used by other segments.

Viewing the High Water Mark

Prior to deallocation, you can use the DBMS_SPACE package, which contains a procedure (UNUSED_SPACE) that returns information about the position of the high water mark and the amount of unused space in a segment.

Within a segment, the high water mark indicates the amount of used space. You cannot release space below the high water mark (even if there is no data in the space you wish to deallocate). However, if the segment is completely empty, you can release space using the TRUNCATE DROP STORAGE statement.

Issuing Space Deallocation Statements

The following statements deallocate unused space in a segment (table, index or cluster). The KEEP clause is *optional*.

```
ALTER TABLE table DEALLOCATE UNUSED KEEP integer;
ALTER INDEX index DEALLOCATE UNUSED KEEP integer;
ALTER CLUSTER cluster DEALLOCATE UNUSED KEEP integer;
```

When you explicitly identify an amount of unused space to KEEP, this space is retained while the remaining unused space is deallocated. If the remaining number of extents becomes smaller than MINEXTENTS, the MINEXTENTS value changes to reflect the new number. If the initial extent becomes smaller, the INITIAL value changes to reflect the new size of the initial extent.

If you do not specify the KEEP clause, all unused space (everything above the high water mark) is deallocated, as long as the size of the initial extent and MINEXTENTS are preserved. Thus, even if the high water mark occurs within the MINEXTENTS boundary, MINEXTENTS remains and the initial extent size is not reduced.

See Also: For details on the syntax and options associated with deallocating unused space, see the *Oracle7 Server SQL Reference.*

You can verify that deallocated space is freed by looking at the DBA_FREE_SPACE view. For more information on this view, see the *Oracle7 Server Reference.*

For details about the DBMS_SPACE package, see page 16 – 25.

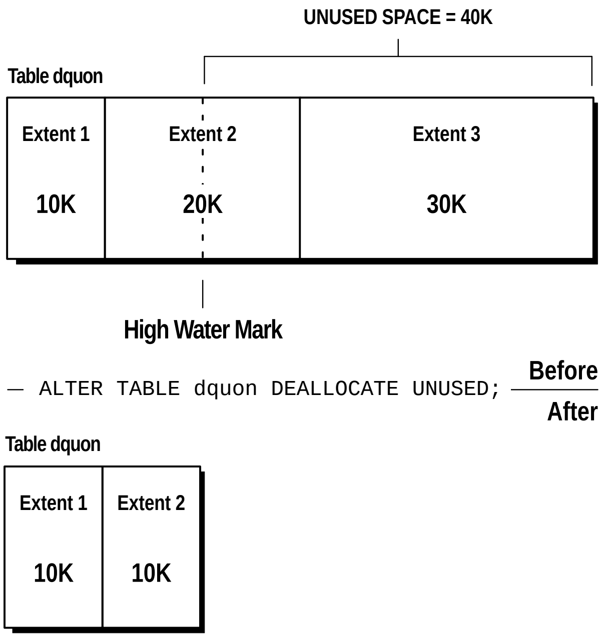
Deallocating Space: Examples

This section includes various space deallocation scenarios. Prior to reading it, you should familiarize yourself with the ALTER...DEALLOCATE UNUSED statements in the *Oracle7 Server SQL Reference.*

Example 1

Table dquon consists of three extents (see figure Figure 10 – 3). The first extent is 10K, the second is 20K, and the third is 30K. The high water mark is in the middle of the second extent, and there is 40K of unused space. The following statement deallocates all unused space, leaving table dquon with two remaining extents. The third extent disappears, and the second extent size is 10K.

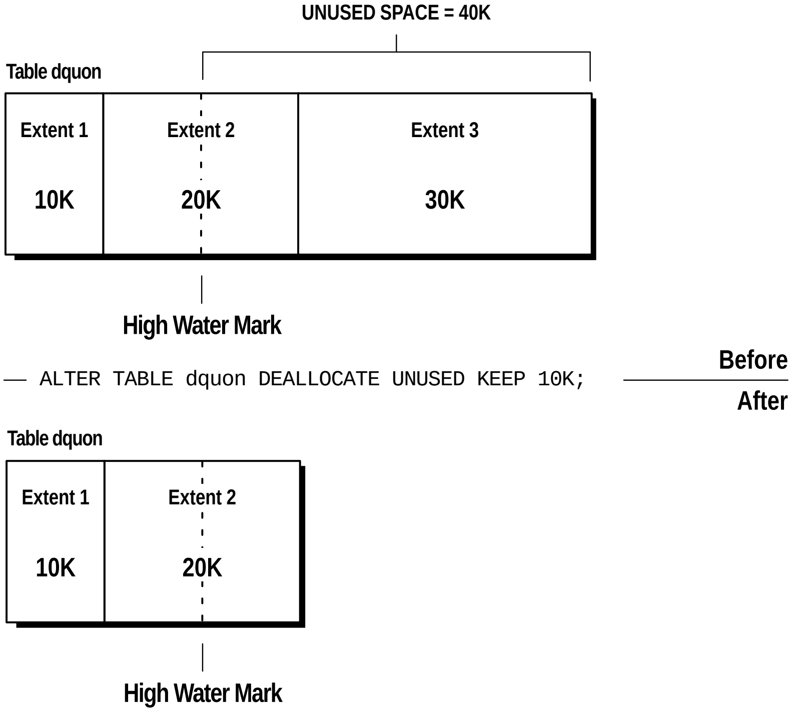
```
ALTER TABLE dquon DEALLOCATE UNUSED;
```



```
— ALTER TABLE dquon DEALLOCATE UNUSED;
```

Figure 10 – 3 Deallocating All Unused Space

If you deallocate all unused space from dquon and KEEP 10K (see Figure 10 – 4), the third extent is deallocated and the second extent remains in tact.



```
ALTER TABLE dquon DEALLOCATE UNUSED KEEP 10K;
```

Figure 10 – 4 Deallocating Unused Space, KEEP 10K

If you deallocate all unused space from dquon and KEEP 20K, the third extent is cut to 10K, and the size of the second extent remains the same.

```
ALTER TABLE dquon DEALLOCATE UNUSED KEEP 20K;
```

Example 2

When you issue the ALTER TABLE dquon DEALLOCATE UNUSED statement, you completely deallocate the third extent, and the second extent is left with 10K. Note that the size of the next allocated extent defaults to the size of the last completely deallocated extent, which in this example, is 30K. However, if you can explicitly set the size of the next extent using the ALTER ... STORAGE [NEXT] statement.

Example 3

To preserve the MINEXTENTS number of extents, DEALLOCATE can retain extents that were originally allocated to an instance (added below the high water mark), while deallocating extents that were originally allocated to the segment.

For example, table dquon has a MINEXTENTS value of 2. Examples 1 and 2 still yield the same results. However, if the MINEXTENTS value is 3, then the ALTER TABLE dquon DEALLOCATE UNUSED statement has no effect, while the ALTER TABLE dquon DEALLOCATE UNUSED KEEP 10K statement removes the third extent and changes the value of MINEXTENTS to 2.

Understanding Space Use of Datatypes

When creating tables and other data structures, you need to know how much space they will require. Each datatype has different space requirements, as described below.

Character Datatypes

The CHAR and VARCHAR2 datatypes store alphanumeric data in strings of ASCII (American Standard Code for Information Interchange) or EBCDIC (Extended Binary Coded Decimal Interchange Code) values, depending on the character set used by the hardware that runs Oracle. Character datatypes can also store data using character sets supported by the National Language Support (NLS) feature of Oracle.

The CHAR datatype stores fixed length character strings. When a table is created with a CHAR column, a column length (in bytes, not characters) between 1 and 255 can be specified for the CHAR column; the default is 1 byte. Extra blanks are used to fill remaining space in the column for values less than the column length.

The VARCHAR2 datatype stores variable length character strings. When a table is created with a VARCHAR2 column, a maximum column length (in bytes, not characters) between 1 and 2000 is specified for the VARCHAR2 column. For each row, each value in the column is stored as a variable length field. Extra blanks are not used to fill remaining space in the column.

Number Datatype

The NUMBER datatype stores fixed and floating point numbers. Positive numbers in the range 1 x 10^–130 to 9.99...9 x 10^125 (with up to 38 significant digits), negative numbers in the range

–1 x 10^–130 to –9.99..9 x 10^125 (with up to 38 significant digits), and zero. You can optionally specify a *precision* (total number of digits) and *scale* (number of digits to the right of the decimal point) when defining a NUMBER column. If precision is not specified, the column stores values as given. If no scale is specified, the scale defaults to zero.

Oracle guarantees portability of numbers with a precision equal to or less than 38 digits. You can specify a scale and no precision:

```
column_name NUMBER (*, scale)
```

In this case, the precision is 38 and the specified scale is maintained.

DATE Datatype

The DATE datatype stores point–in–time values, such as dates and times. Date data is stored in fixed length fields of seven bytes each.

LONG Datatype

Columns defined as LONG store variable length character data containing up to two gigabytes of information. LONG data is text data, and is appropriately converted when moved between different character sets. LONG data cannot be indexed.

RAW and LONG RAW Datatypes

RAW is a variable length datatype like the VARCHAR2 character datatype, except that SQL*Net (which connects users sessions to the instance) and the Import and Export utilities do not perform character conversion when transmitting RAW or LONG RAW data. In contrast, SQL*Net and Export/Import automatically convert CHAR, VARCHAR2, and LONG data between the database character set and the user session character set (set by the NLS_LANGUAGE parameter of the ALTER SESSION command) if the two character sets are different.

LONG RAW data cannot be indexed, while RAW data can be indexed.

ROWIDs and the ROWID Datatype

Every row in a non–clustered table of an Oracle database is assigned a unique *ROWID* that corresponds to the physical address of a row's row piece (or the initial row piece if the row is chained among multiple row pieces).

Each table in an Oracle database internally has a *pseudo–column* named ROWID. This pseudo–column is not evident when listing the structure of a table by executing a SELECT statement, or a DESCRIBE statement using SQL*Plus, but can be retrieved with a SQL query using the reserved word ROWID as a column name.

ROWIDs use a binary representation of the physical address for each row selected. A ROWID's VARCHAR2 hexadecimal representation is divided into three pieces: *block.slot.file*. Here, *block* is the data block within a file that contains the row, relative to its datafile; *row* is the row in the block; and *file* is the datafile that contains the row. A row's assigned ROWID remains unchanged usually. Exceptions occur when the row is exported and imported (using the Import and Export utilities). When a row is deleted from a table (and the encompassing transaction is committed), the deleted row's associated ROWID can be assigned to a row inserted in a subsequent transaction.

MLSLABEL Datatype

Trusted Oracle7 provides one special datatype, called MLSLABEL. You can declare columns of this datatype in standard Oracle, as well as Trusted Oracle7, for compatibility with Trusted Oracle7 applications.

The MLSLABEL datatype stores a variable length tag (two to five bytes) that represents a binary label in the data dictionary. The ALL_LABELS data dictionary view lists all of the labels ever stored in the database.

See Also: For more information about NLS and support for different character sets, see the *Oracle7 Server Reference.*

For more information about MLSLABEL datatypes, see the *Trusted Oracle7 Server Administrator's Guide.*

Summary of Oracle Datatypes

Table 10 – 2 summarizes important information about each Oracle datatype.

Datatype	Description	Column Length (bytes)
CHAR *(size)*	Fixed length character data of length size.	Fixed for every row in the table (with trailing spaces); maximum size is 255 bytes per row, default size is one byte per row. Consider the character set that is used before setting size. (Are you using a one or two byte character set?)
VARCHAR2 *(size)*	Variable length character data. A maximum size must be specified.	Variable for each row, up to 2000 bytes per row. Consider the character set that is used before setting size. (Are you using a one or two byte character set?)
NUMBER (*p, s*)	Variable length numeric data. Maximum precision p and/or scale s is 38.	Variable for each row. The maximum space required for a given column is 21 bytes per row.
DATE	Fixed length date and time data, ranging from January 1, 4712 B.C. to December 31, 4712 A.D. Default format: DD–MON–YY.	Fixed at seven bytes for each row in the table.
LONG	Variable length character data.	Variable for each row in the table up to 2^31 bytes, or two gigabytes, per row.
RAW *(size)*	Variable length raw binary data. A maximum size must be specified.	Variable for each row in the table, up to 255 bytes per row.
LONG RAW	Variable length raw binary data.	Variable for each row in the table, up to 2^31 bytes, or two gigabytes, per row.
ROWID	Binary data representing row addresses.	Fixed at six bytes for each row in the table.
MLSLABEL	Variable length binary data representing OS labels.	Variable for each row in the table, ranging from two to five bytes per row.

Table 10 – 2 Summary of Oracle Datatype Information

CHAPTER

11

Managing Tables

This chapter describes the various aspects of managing tables, and includes the following topics:

- Guidelines for Managing Tables
- Creating Tables
- Altering Tables
- Manually Allocating Storage for a Table
- Dropping Tables

Before attempting tasks described in this chapter, familiarize yourself with the concepts in Chapter 10, “Guidelines for Managing Schema Objects.”

See Also: This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle SNMP Support Reference Guide.*

Guidelines for Managing Tables

This section describes guidelines to follow when managing tables, and includes the following topics:

- Design Tables Before Creating Them
- Specify How Data Block Space Is to Be Used
- Specify Transaction Entry Parameters
- Specify the Location of Each Table
- Parallelize Table Creation
- Consider Creating UNRECOVERABLE Tables
- Estimate Table Size and Set Storage Parameters
- Plan for Large Tables

Use these guidelines to make managing tables as easy as possible.

Design Tables Before Creating Them

Usually, the application developer is responsible for designing the elements of an application, including the tables. A DBA is responsible for setting storage parameters and defining clusters for tables, based on information from the application developer about how the application works and the types of data expected.

Working with your application developer, carefully plan each table so that the following occurs:

- Tables are normalized.
- Each column is of the proper datatype.
- Columns that allow nulls are defined last, to conserve storage space.
- Tables are clustered whenever appropriate, to conserve storage space and optimize performance of SQL statements.

Specify How Data Block Space Is to Be Used

By specifying the PCTFREE and PCTUSED parameters during the creation of each table, you can affect the efficiency of space utilization and amount of space reserved for updates to the current data in the data blocks of a table's data segment.

See Also: For information about specifying PCTFREE and PCTUSED, see "Managing the Space Usage of Data Blocks" on page 10 – 2.

Specify Transaction Entry Parameters

By specifying the INITRANS and MAXTRANS parameters during the creation of each table, you can affect how much space is initially and can ever be allocated for transaction entries in the data blocks of a table's data segment.

See Also: For information about specifying INITRANS and MAXTRANS, see "Setting Storage Parameters" on page 10 – 7.

Specify the Location of Each Table

If you have the proper privileges and tablespace quota, you can create a new table in any tablespace that is currently online. Therefore, you should specify the TABLESPACE option in a CREATE TABLE statement to identify the tablespace that will store the new table.

If you do not specify a tablespace in a CREATE TABLE statement, the table is created in your default tablespace.

When specifying the tablespace to contain a new table, make sure that you understand implications of your selection. By properly specifying a tablespace during the creation of each table, you can:

- increase the performance of the database system
- decrease the time needed for database administration

The following examples show how incorrect storage locations of schema objects can affect a database:

- If users' objects are created in the SYSTEM tablespace, the performance of Oracle can be reduced, since both data dictionary objects and user objects must contend for the same datafiles.
- If an application's associated tables are arbitrarily stored in various tablespaces, the time necessary to complete administrative operations (such as backup and recovery) for that application's data can be increased.

See Also: For information about specifying tablespaces, see "Assigning Tablespace Quotas" on page 19 – 11.

Parallelize Table Creation

If you have the parallel query option installed, you can parallelize the creation of tables created with a subquery in the CREATE TABLE command. Because multiple processes work together to create the table, performance of the table creation can improve.

See Also: For more information about the parallel query option and parallel table creation, see the *Oracle7 Server Tuning* guide.

For information about the CREATE TABLE command, see the *Oracle7 Server SQL Reference.*

Consider Creating UNRECOVERABLE Tables

You can create a table unrecoverably by specifying UNRECOVERABLE when you create a table with a subquery in the CREATE TABLE AS SELECT statement. However, rows inserted afterwards are recoverable. In fact, after the statement is completed, all future statements are fully recoverable.

Creating a table unrecoverably has the following benefits:

- Space is saved in the redo log files.
- The time it takes to create the table is decreased.
- Performance improves for parallel creation of large tables.

In general when creating a table unrecoverably, the relative performance improvement is greater for larger tables than for smaller tables. Creating small tables unrecoverably has little affect on the time it takes to create a table. However, for larger tables the performance improvement can be significant, especially when you are also parallelizing the table creation.

When you create a table unrecoverably the table cannot be recovered from archived logs (because the needed redo log records are not generated for the unrecoverable table creation). Thus, if you cannot afford to lose the table, you should take a backup after the table is created. In some situations, such as for tables that are created for temporary use, this precaution may not be necessary.

Estimate Table Size and Set Storage Parameters

Estimating the sizes of tables before creating them is useful for the following reasons:

- You can use the combined estimated size of tables, along with estimates for indexes, rollback segments, and redo log files, to determine the amount of disk space that is required to hold an intended database. From these estimates, you can make correct hardware purchases and other decisions.
- You can use the estimated size of an individual table to better manage the disk space that the table will use. When a table is created, you can set appropriate storage parameters and improve I/O performance of applications that use the table.

  For example, assume that you estimate the maximum size of a table before creating it. If you then set the storage parameters when you create the table, fewer extents will be allocated for the table's data segment, and all of the table's data will be stored in a relatively contiguous section of disk space. This decreases the time necessary for disk I/O operations involving this table.

Appendix A contains equations that help estimate the size of tables. Whether or not you estimate table size before creation, you can explicitly set storage parameters when creating each non–clustered table. (Clustered tables automatically use the storage parameters of the cluster.) Any storage parameter that you do not explicitly set when creating or subsequently altering a table automatically uses the corresponding default storage parameter set for the tablespace in which the table resides.

If you explicitly set the storage parameters for the extents of a table's data segment, try to store the table's data in a small number of large extents rather than a large number of small extents.

Plan for Large Tables

There are no limits on the physical size of tables and extents. You can specify the keyword UNLIMITED for MAXEXTENTS, thereby simplifying your planning for large objects, reducing wasted space and fragmentation, and improving space reuse. However, keep in mind that while Oracle allows an unlimited number of extents, when the number of extents in a table grows very large, you may see an impact on performance when performing any operation requiring that table.

> **Note:** You cannot alter data dictionary tables to have MAXEXTENTS greater than the allowed block maximum.

If you have such tables in your database, consider the following recommendations:

Separate the Table from Its Indexes Place indexes in separate tablespaces from other objects, and on separate disks if possible. If you ever need to drop and re–create an index on a very large table (such as when disabling and enabling a constraint, or re–creating the table), indexes isolated into separate tablespaces can often find contiguous space more easily than those in tablespaces that contain other objects.

Allocate Sufficient Temporary Space If applications that access the data in a very large table perform large sorts, ensure that enough space is available for large temporary segments and that users have access to this space. (Note that temporary segments always use the default STORAGE settings for their tablespaces.)

Creating Tables

To create a new table in your schema, you must have the CREATE TABLE system privilege. To create a table in another user's schema, you must have the CREATE ANY TABLE system privilege. Additionally, the owner of the table must have a quota for the tablespace that contains the table, or the UNLIMITED TABLESPACE system privilege.

Create tables using the SQL command CREATE TABLE. When user SCOTT issues the following statement, he creates a non–clustered table named EMP in his schema and stores it in the USERS tablespace:

```
CREATE TABLE emp (
   empno     NUMBER(5) PRIMARY KEY,
   ename     VARCHAR2(15) NOT NULL,
   job       VARCHAR2(10),
   mgr       NUMBER(5),
   hiredate  DATE DEFAULT (sysdate),
   sal       NUMBER(7,2),
   comm      NUMBER(7,2),
   deptno    NUMBER(3) NOT NULL
             CONSTRAINT dept_fkey REFERENCES dept)
   PCTFREE 10
   PCTUSED 40
   TABLESPACE users
   STORAGE ( INITIAL 50K
             NEXT 50K
             MAXEXTENTS 10
             PCTINCREASE 25 );
```

Notice that integrity constraints are defined on several columns of the table and that several storage settings are explicitly specified for the table.

See Also: For more information about system privileges, see Chapter 20. For more information about tablespace quotas, see Chapter 19.

Altering Tables

To alter a table, the table must be contained in your schema, or you must have either the ALTER object privilege for the table or the ALTER ANY TABLE system privilege.

A table in an Oracle database can be altered for the following reasons:

- to add one or more new columns to the table
- to add one or more integrity constraints to a table
- to modify an existing column's definition (datatype, length, default value, and NOT NULL integrity constraint)
- to modify data block space usage parameters (PCTFREE, PCTUSED)
- to modify transaction entry settings (INITRANS, MAXTRANS)
- to modify storage parameters (NEXT, PCTINCREASE)
- to enable or disable integrity constraints or triggers associated with the table
- to drop integrity constraints associated with the table

You can increase the length of an existing column. However, you cannot decrease it unless there are no rows in the table. Furthermore, if you are modifying a table to increase the length of a column of datatype CHAR, realize that this may be a time consuming operation and may require substantial additional storage, especially if the table contains many rows. This is because the CHAR value in each row must be blank–padded to satisfy the new column length.

When altering the data block space usage parameters (PCTFREE and PCTUSED) of a table, note that new settings apply to all data blocks used by the table, including blocks already allocated and subsequently allocated for the table. However, the blocks already allocated for the table are not immediately reorganized when space usage parameters are altered, but as necessary after the change.

When altering the transaction entry settings (INITRANS, MAXTRANS) of a table, note that a new setting for INITRANS only applies to data blocks subsequently allocated for the table, while a new setting for MAXTRANS applies to all blocks (already and subsequently allocated blocks) of a table.

The storage parameters INITIAL and MINEXTENTS cannot be altered. All new settings for the other storage parameters (for example, NEXT, PCTINCREASE) affect only extents subsequently allocated for the table. The size of the next extent allocated is determined by the current values of NEXT and PCTINCREASE, and is not based on previous values of these parameters.

You can alter a table using the SQL command ALTER TABLE. The following statement alters the EMP table:

```
ALTER TABLE emp
   PCTFREE 30
   PCTUSED 60;
```

⚠ **Warning:** Before altering a table, familiarize yourself with the consequences of doing so:

- If a new column is added to a table, the column is initially null. You can add a column with a NOT NULL constraint to a table only if the table does not contain any rows.
- If a view or PL/SQL program unit depends on a base table, the alteration of the base table may affect the dependent object.

See Also: See page 16 – 18 for information about how Oracle manages dependencies.

Manually Allocating Storage for a Table

Oracle dynamically allocates additional extents for the data segment of a table, as required. However, you might want to allocate an additional extent for a table explicitly. For example, when using the Oracle Parallel Server, an extent of a table can be allocated explicitly for a specific instance.

A new extent can be allocated for a table using the SQL command ALTER TABLE with the ALLOCATE EXTENT option.

See Also: For information about the ALLOCATE EXTENT option, see the *Oracle7 Parallel Server Concepts & Administration* guide.

Dropping Tables

To drop a table, the table must be contained in your schema or you must have the DROP ANY TABLE system privilege.

To drop a table that is no longer needed, use the SQL command DROP TABLE. The following statement drops the EMP table:

```
DROP TABLE emp;
```

If the table to be dropped contains any primary or unique keys referenced by foreign keys of other tables and you intend to drop the FOREIGN KEY constraints of the child tables, include the CASCADE option in the DROP TABLE command, as shown below:

```
DROP TABLE emp CASCADE CONSTRAINTS;
```

⚠ **Warning:** Before dropping a table, familiarize yourself with the consequences of doing so:

- Dropping a table removes the table definition from the data dictionary. All rows of the table are no longer accessible.
- All indexes and triggers associated with a table are dropped.
- All views and PL/SQL program units dependent on a dropped table remain, yet become invalid (not usable). See page 16 – 18 for information about how Oracle manages such dependencies.
- All synonyms for a dropped table remain, but return an error when used.
- All extents allocated for a non–clustered table that is dropped are returned to the free space of the tablespace and can be used by any other object requiring new extents or new objects.
- All rows corresponding to a clustered table are deleted from the blocks of the cluster.

CHAPTER

12

Managing Views, Sequences and Synonyms

This chapter describes aspects of view management, and includes the following topics:

- Managing Views
- Managing Sequences
- Managing Synonyms

Before attempting tasks described in this chapter, familiarize yourself with the concepts in Chapter 10, “Guidelines for Managing Schema Objects.”

See Also: This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User’s Guide*.

Managing Views

A view is a tailored presentation of the data contained in one or more tables (or other views), and takes the output of a query and treats it as a table. You can think of a view as a "stored query" or a "virtual table." You can use views in most places where a table can be used.

This section describes aspects of managing views, and includes the following topics:

- Creating Views
- Modifying a Join View
- Replacing Views
- Dropping Views

Creating Views

To create a view, you must fulfill the requirements listed below:

- To create a view in your schema, you must have the CREATE VIEW privilege; to create a view in another user's schema, you must have the CREATE ANY VIEW system privilege. You may acquire these privileges explicitly or via a role.
- The *owner* of the view (whether it is you or another user) must have been explicitly granted privileges to access all objects referenced in the view definition; the owner *cannot* have obtained these privileges through roles. Also, the functionality of the view is dependent on the privileges of the view's owner. For example, if the owner of the view has only the INSERT privilege for Scott's EMP table, the view can only be used to insert new rows into the EMP table, not to SELECT, UPDATE, or DELETE rows from it.
- If the owner of the view intends to grant access to the view to other users, the owner must have received the object privileges to the base objects with the GRANT OPTION or the system privileges with the ADMIN OPTION.

You can create views using the SQL command CREATE VIEW. Each view is defined by a query that references tables, snapshots, or other views. The query that defines a view cannot contain the ORDER BY or FOR UPDATE clauses. For example, the following statement creates a view on a subset of data in the EMP table:

```
CREATE VIEW sales_staff AS
   SELECT empno, ename, deptno
   FROM emp
   WHERE deptno = 10
   WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

The query that defines the SALES_STAFF view references only rows in department 10. Furthermore, the CHECK OPTION creates the view with the constraint that INSERT and UPDATE statements issued against the view cannot result in rows that the view cannot select. For example, the following INSERT statement successfully inserts a row into the EMP table by means of the SALES_STAFF view, which contains all rows with department number 10:

```
INSERT INTO sales_staff VALUES (7584, ’OSTER’, 10);
```

However, the following INSERT statement is rolled back and returns an error because it attempts to insert a row for department number 30, which could not be selected using the SALES_STAFF view:

```
INSERT INTO sales_staff VALUES (7591, ’WILLIAMS’, 30);
```

The following statement creates a view that joins data from the EMP and DEPT tables:

```
CREATE VIEW division1_staff AS
   SELECT ename, empno, job, dname
   FROM emp, dept
   WHERE emp.deptno IN (10, 30)
   AND emp.deptno = dept.deptno;
```

The DIVISION1_STAFF view joins information from the EMP and DEPT tables. The CHECK OPTION is not specified in the CREATE VIEW statement for this view.

Expansion of Defining Queries at View Creation Time

In accordance with the ANSI / ISO standard, Oracle expands any wildcard in a top–level view query into a column list when a view is created, and stores the resulting query in the data dictionary; any subqueries are left intact. The column names in an expanded column list are enclosed in quote marks to account for the possibility that the columns of the base object were originally entered with quotes and require them for the query to be syntactically correct.

As an example, assume that the DEPT view is created as follows:

```
CREATE VIEW dept AS SELECT * FROM scott.dept;
```

Oracle stores the defining query of the DEPT view as:

```
SELECT ”DEPTNO”, ”DNAME”, ”LOC” FROM scott.dept
```

Views created with errors do not have wildcards expanded. However, if the view is eventually compiled without errors, wildcards in the defining query are expanded.

Creating Views with Errors

If there are no syntax errors in a CREATE VIEW statement, Oracle can create the view even if the defining query of the view cannot be executed; the view is considered “created with errors.” For example, when a view is created that refers to a non–existent table or an invalid column of an existing table, or when the view owner does not have the required privileges, the view can be created anyway and entered into the data dictionary. However, the view is not yet usable.

To create a view with errors, you must include the FORCE option of the CREATE VIEW command:

```
CREATE FORCE VIEW AS ....;
```

By default, views are not created with errors. When a view is created with errors, Oracle returns a message indicating the view was created with errors. The status of a view created with errors is INVALID. If conditions later change so that the query of an invalid view can be executed, the view can be recompiled and become valid (usable).

See Also: For information changing conditions and their impact on views, see “Managing Object Dependencies” on page 16 – 18.

Modifying a Join View

A *modifiable join view* is a view that contains more than one table in the top–level FROM clause of the SELECT statement, and that does *not* contain any of the following:

- DISTINCT operator
- aggregate functions: AVG, COUNT, GLB, MAX, MIN, STDDEV, SUM, or VARIANCE
- set operations: UNION, UNION ALL, INTERSECT, MINUS
- GROUP BY or HAVING clauses
- START WITH or CONNECT BY clauses
- ROWNUM pseudocolumn

With some restrictions, you can modify views that involve joins. If a view is a join on other nested views, then the other nested views must be mergeable into the top level view.

The examples in following sections use the EMP and DEPT tables. These examples work only if you explicitly define the primary and foreign keys in these tables, or define unique indexes. Following are the appropriately constrained table definitions for EMP and DEPT:

```
CREATE TABLE dept (
        deptno   NUMBER(4) PRIMARY KEY,
        dname    VARCHAR2(14),
        loc      VARCHAR2(13));

CREATE TABLE emp (
        empno    NUMBER(4) PRIMARY KEY,
        ename    VARCHAR2(10),
        job      varchar2(9),
        mgr      NUMBER(4),
        hiredate DATE,
        sal      NUMBER(7,2),
        comm     NUMBER(7,2),
        deptno   NUMBER(2),
        FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO));.
```

You could also omit the primary and foreign key constraints listed above, and create a UNIQUE INDEX on DEPT (DEPTNO) to make the following examples work.

See Also: For more information about mergeable views see Chapter 5 in the *Oracle7 Server Tuning* manual.

Key–Preserved Tables

The concept of a *key–preserved table* is fundamental to understanding the restrictions on modifying join views. A table is key preserved if every key of the table can also be a key of the result of the join. So, a key–preserved table has its keys preserved through a join.

> **Note:** It is not necessary that the key or keys of a table be selected for it to be key preserved. It is sufficient that if the key or keys were selected, then they would also be key(s) of the result of the join.

☞ **Attention:** The key–preserving property of a table does not depend on the actual data in the table. It is, rather, a property of its schema and not of the data in the table. For example, if in the EMP table there was at most one employee in each department, then DEPT.DEPTNO would be unique in the result of a join of EMP and DEPT, but DEPT would still not be a key–preserved table.

If you SELECT all rows from EMP_DEPT_VIEW, the results are:

```
EMPNO      ENAME      DEPTNO     DNAME          LOC
---------- ---------- ---------- -------------- -----
      7782 CLARK              10 ACCOUNTING     NEW YORK
      7839 KING               10 ACCOUNTING     NEW YORK
      7934 MILLER             10 ACCOUNTING     NEW YORK
      7369 SMITH              20 RESEARCH       DALLAS
      7876 ADAMS              20 RESEARCH       DALLAS
      7902 FORD               20 RESEARCH       DALLAS
      7788 SCOTT              20 RESEARCH       DALLAS
      7566 JONES              20 RESEARCH       DALLAS
8 rows selected.
```

In this view, EMP is a key–preserved table, because EMPNO is a key of the EMP table, and also a key of the result of the join. DEPT is *not* a key–preserved table, because although DEPTNO is a key of the DEPT table, it is not a key of the join.

DML Statements and Join Views

Any UPDATE, INSERT, or DELETE statement on a join view can modify only one underlying base table.

UPDATE Statements The following example shows an UPDATE statement that successfully modifies the EMP_DEPT view:

```
UPDATE emp_dept
  SET sal = sal * 1.10
    WHERE deptno = 10;
```

The following UPDATE statement would be disallowed on the EMP_DEPT view:

```
UPDATE emp_dept
  SET loc = ’BOSTON’
    WHERE ename = ’SMITH’;
```

This statement fails with an ORA–01779 error (“cannot modify a column which maps to a non key–preserved table”), because it attempts to modify the underlying DEPT table, and the DEPT table is not key preserved in the EMP_DEPT view.

In general, all modifiable columns of a join view must map to columns of a key–preserved table. If the view is defined using the WITH CHECK OPTION clause, then all join columns and all columns of repeated tables are not modifiable.

So, for example, if the EMP_DEPT view were defined using WITH CHECK OPTION, the following UPDATE statement would fail:

```
UPDATE emp_dept
  SET deptno = 10
    WHERE ename = ’SMITH’;
```

The statement fails because it is trying to update a join column.

DELETE Statements You can delete from a join view provided there is *one and only one* key–preserved table in the join.

The following DELETE statement works on the EMP_DEPT view:

```
DELETE FROM emp_dept
  WHERE ename = ’SMITH’;
```

This DELETE statement on the EMP_DEPT view is legal because it can be translated to a DELETE operation on the base EMP table, and because the EMP table is the only key–preserved table in the join.

In the following view, a DELETE operation cannot be performed on the view because both E1 and E2 are key–preserved tables:

```
CREATE VIEW emp_emp AS
  SELECT e1.ename, e2.empno, deptno
    FROM emp e1, emp e2
    WHERE e1.empno = e2.empno;
```

If a view is defined using the WITH CHECK OPTION clause and the key–preserved table is repeated, then rows cannot be deleted from such a view:

```
CREATE VIEW emp_mgr AS
  SELECT e1.ename, e2.ename mname
    FROM emp e1, emp e2
      WHERE e1.mgr = e2.empno
      WITH CHECK OPTION;
```

No deletion can be performed on this view because the view involves a self–join of the table that is key preserved.

INSERT Statements The following INSERT statement on the EMP_DEPT view succeeds:

```
INSERT INTO emp_dept (ename, empno, deptno)
  VALUES (’KURODA’, 9010, 40);
```

This statement works because only one key–preserved base table is being modified (EMP), and 40 is a valid DEPTNO in the DEPT table (thus satisfying the FOREIGN KEY integrity constraint on the EMP table).

An INSERT statement like the following would fail for the same reason that such an UPDATE on the base EMP table would fail: the FOREIGN KEY integrity constraint on the EMP table is violated.

```
INSERT INTO emp_dept (ename, empno, deptno)
  VALUES (’KURODA’, 9010, 77);
```

The following INSERT statement would fail with an ORA–01776 error (“cannot modify more than one base table through a view”).

```
INSERT INTO emp_dept (empno, ename, loc)
  VALUES (9010, ’KURODA’, ’BOSTON’);
```

An INSERT cannot, implicitly or explicitly, refer to columns of a non–key–preserved table. If the join view is defined using the WITH CHECK OPTION clause, then you cannot perform an INSERT to it.

Using the UPDATABLE_ COLUMNS Views

The views described in Table 12 – 1 can assist you when modifying join views.

View Name	Description
USER_UPDATABLE_COLUMNS	Shows all columns in all tables and views in the user’s schema that are modifiable.
DBA_UPDATABLE_COLUMNS	Shows all columns in all tables and views in the DBA schema that are modifiable.
ALL_UPDATABLE_VIEWS	Shows all columns in all tables and views that are modifiable.

Table 12 – 1 UPDATABLE_COLUMNS Views

Replacing Views

To replace a view, you must have all the privileges required to drop and create a view. If the definition of a view must change, the view must be replaced; you cannot alter the definition of a view. You can replace views in the following ways:

- You can drop and re–create the view.

⚠ **Warning:** When a view is dropped, all grants of corresponding object privileges are revoked from roles and users. After the view is re–created, privileges must be re–granted.

- You can redefine the view with a CREATE VIEW statement that contains the OR REPLACE option. The OR REPLACE option replaces the current definition of a view and preserves the current security authorizations. For example, assume that you create the SALES_STAFF view as given in the previous example, and grant several object privileges to roles and other users. However, now you need to redefine the SALES_STAFF view to change the department number specified in the WHERE clause. You can replace the current version of the SALES_STAFF view with the following statement:

```
CREATE OR REPLACE VIEW sales_staff AS
   SELECT empno, ename, deptno
   FROM emp
   WHERE deptno = 30
   WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

Before replacing a view, consider the following effects:

- Replacing a view replaces the view's definition in the data dictionary. All underlying objects referenced by the view are not affected.
- If a constraint in the CHECK OPTION was previously defined but not included in the new view definition, the constraint is dropped.
- All views and PL/SQL program units dependent on a replaced view become invalid (not usable). See page 16 – 18 for more information on how Oracle manages such dependencies.

Dropping Views

You can drop any view contained in your schema. To drop a view in another user's schema, you must have the DROP ANY VIEW system privilege. Drop a view using the SQL command DROP VIEW. For example, the following statement drops a view named SALES_STAFF:

```
DROP VIEW sales_staff;
```

Managing Sequences

This section describes various aspects of managing sequences, and includes the following topics:

- Creating Sequences
- Altering Sequences
- Initialization Parameters Affecting Sequences
- Dropping Sequences

Creating Sequences

To create a sequence in your schema, you must have the CREATE SEQUENCE system privilege; to create a sequence in another user's schema, you must have the CREATE ANY SEQUENCE privilege. Create a sequence using the SQL command CREATE SEQUENCE. For example, the following statement creates a sequence used to generate employee numbers for the EMPNO column of the EMP table:

```
CREATE SEQUENCE emp_sequence
   INCREMENT BY 1
   START WITH 1
   NOMAXVALUE
   NOCYCLE
   CACHE 10;
```

The CACHE option pre–allocates a set of sequence numbers and keeps them in memory so that sequence numbers can be accessed faster. When the last of the sequence numbers in the cache has been used, Oracle reads another set of numbers into the cache.

Oracle might skip sequence numbers if you choose to cache a set of sequence numbers. For example, when an instance abnormally shuts down (for example, when an instance failure occurs or a SHUTDOWN ABORT statement is issued), sequence numbers that have been cached but not used are lost. Also, sequence numbers that have been used but not saved are lost as well. Oracle might also skip cached sequence numbers after an export and import; see the *Oracle7 Server Utilities* guide for details.

See Also: For information about how the Oracle Parallel Server affects cached sequence numbers, see the *Oracle7 Parallel Server Concepts & Administration* guide.

For performance information on caching sequence numbers, see the *Oracle7 Server Tuning* manual.

Altering Sequences

To alter a sequence, your schema must contain the sequence, or you must have the ALTER ANY SEQUENCE system privilege. You can alter a sequence to change any of the parameters that define how it generates sequence numbers except the sequence's starting number. To change the starting point of a sequence, drop the sequence and then re–create it.

Alter a sequence using the SQL command ALTER SEQUENCE. For example, the following statement alters the EMP_SEQUENCE:

```
ALTER SEQUENCE emp_sequence
   INCREMENT BY 10
   MAXVALUE 10000
   CYCLE
   CACHE 20;
```

Initialization Parameters Affecting Sequences

The initialization parameter SEQUENCE_CACHE_ENTRIES sets the number of sequences that may be cached at any time. If auditing is enabled for your system, allow one additional sequence for the sequence to identify audit session numbers.

If the value for SEQUENCE_CACHE_ENTRIES is too low, Oracle might skip sequence values, as in the following scenario: assume you are using five cached sequences, the cache is full, and SEQUENCE_CACHE_ENTRIES = 4. If four sequences are currently cached, then a fifth sequence replaces the least recently used sequence in the cache and all remaining values (up to the last sequence number cached) in the displaced sequence are lost.

Dropping Sequences

You can drop any sequence in your schema. To drop a sequence in another schema, you must have the DROP ANY SEQUENCE system privilege. If a sequence is no longer required, you can drop the sequence using the SQL command DROP SEQUENCE. For example, the following statement drops the ORDER_SEQ sequence:

```
DROP SEQUENCE order_seq;
```

When a sequence is dropped, its definition is removed from the data dictionary. Any synonyms for the sequence remain, but return an error when referenced.

Managing Synonyms

You can create both public and private synonyms. A *public* synonym is owned by the special user group named PUBLIC and is accessible to every user in a database. A *private* synonym is contained in the schema of a specific user and available only to the user and the user's grantees.

This section includes the following synonym management information:

- Creating Synonyms
- Dropping Synonyms

Creating Synonyms

To create a private synonym in your own schema, you must have the CREATE SYNONYM privilege; to create a private synonym in another user's schema, you must have the CREATE ANY SYNONYM privilege. To create a public synonym, you must have the CREATE PUBLIC SYNONYM system privilege.

Create a synonym using the SQL command CREATE SYNONYM. For example, the following statement creates a public synonym named PUBLIC_EMP on the EMP table contained in the schema of JWARD:

```
CREATE PUBLIC SYNONYM public_emp FOR jward.emp;
```

Dropping Synonyms

You can drop any private synonym in your own schema. To drop a private synonym in another user's schema, you must have the DROP ANY SYNONYM system privilege. To drop a public synonym, you must have the DROP PUBLIC SYNONYM system privilege.

Drop a synonym that is no longer required using the SQL command DROP SYNONYM. To drop a private synonym, omit the PUBLIC keyword; to drop a public synonym, include the PUBLIC keyword.

For example, the following statement drops the private synonym named EMP:

```
DROP SYNONYM emp;
```

The following statement drops the public synonym named PUBLIC_EMP:

```
DROP PUBLIC SYNONYM public_emp;
```

When you drop a synonym, its definition is removed from the data dictionary. All objects that reference a dropped synonym remain; however, they become invalid (not usable).

See Also: For more information about how dropping synonyms can affect other schema objects, see "Managing Object Dependencies" on page 16 – 18.

CHAPTER

13 Managing Indexes

This chapter describes various aspects of index management, and includes the following topics:

- Guidelines for Managing Indexes
- Creating Indexes
- Altering Indexes
- Monitoring Space Use of Indexes
- Dropping Indexes

Before attempting tasks described in this chapter, familiarize yourself with the concepts in Chapter 10, “Guidelines for Managing Schema Objects.”

Guidelines for Managing Indexes

This section describes guidelines to follow when managing indexes, and includes the following topics:

- Create Indexes After Inserting Table Data
- Limit the Number of Indexes per Table
- Specify Transaction Entry Parameters
- Specify Index Block Space Use
- Specify the Tablespace for Each Index
- Parallelize Index Creation
- Consider Creating UNRECOVERABLE Indexes
- Estimate Index Size and Set Storage Parameters

An *index* is an optional structure associated with tables and clusters, which you can create explicitly to speed SQL statement execution on a table. Just as the index in this manual helps you locate information faster than if there were no index, an Oracle index provides a faster access path to table data.

The absence or presence of an index does not require a change in the wording of any SQL statement. An index merely offers a fast access path to the data; it affects only the speed of execution. Given a data value that has been indexed, the index points directly to the location of the rows containing that value.

Indexes are logically and physically independent of the data in the associated table. You can create or drop an index at anytime without effecting the base tables or other indexes. If you drop an index, all applications continue to work; however, access of previously indexed data might be slower. Indexes, as independent structures, require storage space.

Oracle automatically maintains and uses indexes after they are created. Oracle automatically reflects changes to data, such as adding new rows, updating rows, or deleting rows, in all relevant indexes with no additional action by users.

See Also: For information about performance implications of index creation, see the *Oracle7 Server Tuning* manual.

For more information about indexes, see the *Oracle7 Server Concepts* guide.

Create Indexes After Inserting Table Data

You should create an index for a table after inserting or loading data (via SQL*Loader or Import) into the table. It is more efficient to insert rows of data into a table that has no indexes and then create the indexes for subsequent access. If you create indexes before table data is loaded, every index must be updated every time a row is inserted into the table. You should also create the index for a cluster before inserting any data into the cluster.

When an index is created on a table that already has data, Oracle must use sort space. Oracle uses the sort space in memory allocated for the creator of the index (the amount per user is determined by the initialization parameter SORT_AREA_SIZE), but must also swap sort information to and from temporary segments allocated on behalf of the index creation.

If the index is extremely large, you may want to perform the following tasks:

To Manage a Large Index

1. Create a new temporary segment tablespace.
2. Alter the index creator's temporary segment tablespace.
3. Create the index.
4. Remove the temporary segment tablespace and re–specify the creator's temporary segment tablespace, if desired.

See Also: Under certain conditions, data can be loaded into a table with SQL*Loader's "direct path load" and an index can be created as data is loaded; see the *Oracle7 Server Utilities* guide for more information.

Limit the Number of Indexes per Table

A table can have any number of indexes. However, the more indexes there are, the more overhead is incurred as the table is modified. Specifically, when rows are inserted or deleted, all indexes on the table must be updated as well. Also, when a column is updated, all indexes that contain the column must be updated.

Thus, there is a tradeoff between the speed of retrieving data from a table and the speed of updating the table. For example, if a table is primarily read–only, having more indexes can be useful, but if a table is heavily updated, having fewer indexes may be preferable.

Specify Transaction Entry Parameters

By specifying the INITRANS and MAXTRANS parameters during the creation of each index, you can affect how much space is initially and can ever be allocated for transaction entries in the data blocks of an index's segment.

See Also: For more information about setting these parameters, see "Setting Storage Parameters" on page 10 – 7.

Specify Index Block Space Use

When an index is created for a table, data blocks of the index are filled with the existing values in the table up to PCTFREE. The space reserved by PCTFREE for an index block is only used when a new row is inserted into the table and the corresponding index entry must be placed in the correct index block (that is, between preceding and following index entries); if no more space is available in the appropriate index block, the indexed value is placed in another index block. Therefore, if you plan on inserting many rows into an indexed table, PCTFREE should be high to accommodate the new index values; if the table is relatively static without many inserts, PCTFREE for an associated index can be low so that fewer blocks are required to hold the index data.

See Also: PCTUSED cannot be specified for indexes. See "Managing the Space Usage of Data Blocks" on page 10 – 2 for information about the PCTFREE parameter.

Specify the Tablespace for Each Index

Indexes can be created in any tablespace. An index can be created in the same or different tablespace as the table it indexes.

If you use the same tablespace for a table and its index, then database maintenance may be more convenient (such as tablespace or file backup and application availability or update) and all the related data will always be online together.

Using different tablespaces (on different disks) for a table and its index produces better performance than storing the table and index in the same tablespace, due to reduced disk contention.

If you use different tablespaces for a table and its index and one tablespace is offline (containing either data or index), then the statements referencing that table are not guaranteed to work.

Parallelize Index Creation

If you have the parallel query option installed, you can parallelize index creation. Because multiple processes work together to create the index, Oracle can create the index more quickly than if a single server process created the index sequentially.

When creating an index in parallel, storage parameters are used separately by each query server process. Therefore, an index created with an INITIAL of 5M and a PARALLEL DEGREE of 12 consumes at least 60M of storage during index creation.

See Also: For more information on the parallel query option and parallel index creation, see the *Oracle7 Server Tuning* manual.

Consider Creating UNRECOVERABLE Indexes

You can create an index without generating any redo log records by specifying UNRECOVERABLE in the CREATE INDEX statement.

> **Note:** Because indexes created unrecoverably are not archived, you should perform a backup after you create the index.

Creating an index unrecoverably has the following benefits:

- Space is saved in the redo log files.
- The time it takes to create the index is decreased.
- Performance improves for parallel creation of large indexes.

In general when creating an index unrecoverably, the relative performance improvement is greater for larger indexes than for smaller ones. Creating small indexes unrecoverably has little affect on the time it takes to create an index. However, for larger indexes the performance improvement can be significant, especially when you are also parallelizing the index creation.

Estimate Index Size and Set Storage Parameters

Appendix A contains equations that help estimate the size of indexes.

Estimating the size of an index before creating one is useful for the following reasons:

- You can use the combined estimated size of indexes, along with estimates for tables, rollback segments, and redo log files, to determine the amount of disk space that is required to hold an intended database. From these estimates, you can make correct hardware purchases and other decisions.
- You can use the estimated size of an individual index to better manage the disk space that the index will use. When an index is created, you can set appropriate storage parameters and improve I/O performance of applications that use the index.

  For example, assume that you estimate the maximum size of a table before creating it. If you then set the storage parameters when you create the table, fewer extents will be allocated for the table’s data segment, and all of the table’s data will be stored in a relatively contiguous section of disk space. This decreases the time necessary for disk I/O operations involving this table.

The maximum size of a single index entry is roughly one–half the data block size minus some overhead.

As with tables, you can explicitly set storage parameters when creating an index. If you explicitly set the storage parameters for an index, try to store the index's data in a small number of large extents rather than a large number of small extents.

See Also: For specific information about storage parameters, see "Setting Storage Parameters" on page 10 – 7.

For specific information about estimating index size, see Appendix A.

Considerations Before Disabling or Dropping Constraints

Because unique and primary keys have associated indexes, you should factor in the cost of dropping and creating indexes when considering whether to disable or drop a UNIQUE or PRIMARY KEY constraint. If the associated index for a UNIQUE key or PRIMARY KEY constraint is extremely large, you may save time by leaving the constraint enabled rather than dropping and re–creating the large index.

Creating Indexes

This section describes how to create an index, and includes the following topics:

- Creating an Index Associated with a Constraint
- Creating an Index Explicitly
- Re–Creating an Existing Index

Before you can create a new index you must own or have the INDEX object privilege for the corresponding table. The schema that contains the index must also have a quota for the tablespace intended to contain the index, or the UNLIMITED TABLESPACE system privilege. To create an index in another user's schema, you must have the CREATE ANY INDEX system privilege.

To enable a UNIQUE key or PRIMARY KEY (which creates an associated index), the owner of the table needs a quota for the tablespace intended to contain the index, or the UNLIMITED TABLESPACE system privilege.

LONG and LONG RAW columns cannot be indexed.

Oracle enforces a UNIQUE key or PRIMARY KEY integrity constraint by creating a unique index on the unique key or primary key. This index is automatically created by Oracle when the constraint is enabled; no action is required by the issuer of the CREATE TABLE or ALTER TABLE statement to create the index. This includes both when a constraint is defined and enabled, and when a defined but disabled constraint is enabled.

In general, it is better to create constraints to enforce uniqueness than it is to use the CREATE UNIQUE INDEX syntax. A constraint's associated index always assumes the name of the constraint; you cannot specify a specific name for a constraint index.

If you do not specify the storage options for an index, they are automatically set to the default storage options of the host tablespace.

Creating an Index Associated with a Constraint

You can set the storage options for the indexes associated with UNIQUE key and PRIMARY KEY constraints using the ENABLE clause with the USING INDEX option. The following statement defines a PRIMARY KEY constraint and specifies the associated index's storage option:

```
CREATE TABLE emp (
   empno NUMBER(5) PRIMARY KEY, . . . )
   ENABLE PRIMARY KEY USING INDEX
      TABLESPACE users
      PCTFREE 0;
```

Creating an Index Explicitly

You can create indexes explicitly (outside of integrity constraints) using the SQL command CREATE INDEX. The following statement creates an index named EMP_ENAME for the ENAME column of the EMP table:

```
CREATE INDEX emp_ename ON emp(ename)
   TABLESPACE users
   STORAGE (INITIAL 20K
      NEXT 20k
      PCTINCREASE 75)
   PCTFREE 0;
```

Notice that several storage settings are explicitly specified for the index.

Re–Creating an Existing Index

You can create an index using an existing index as the data source. Creating an index in this manner allows you to change storage characteristics, or move to a new tablespace. Re–creating an index based on an existing data source also removes intra–block fragmentation. In fact, compared to dropping the index and using the CREATE INDEX command, re–creating an existing index offers better performance.

Issue the following statement to re–create an existing index:

```
ALTER INDEX index name REBUILD;
```

The REBUILD clause must immediately follow the index name, and precede any other options. Also, the REBUILD clause cannot be used in conjunction with the DEALLOCATE STORAGE clause.

See Also: For more information on the ALTER INDEX command and optional clauses, see the *Oracle7 Server SQL Reference.*

Altering Indexes

To alter an index, your schema must contain the index or you must have the ALTER ANY INDEX system privilege. You can alter an index only to change the transaction entry parameters or to change the storage parameters; you cannot change its column structure.

Alter the storage parameters of any index, including those created by Oracle to enforce primary and unique key integrity constraints, using the SQL command ALTER INDEX. For example, the following statement alters the EMP_ENAME index:

```
ALTER INDEX emp_ename
   INITRANS 5
   MAXTRANS 10
   STORAGE (PCTINCREASE 50);
```

When you alter the transaction entry settings (INITRANS, MAXTRANS) of an index, a new setting for INITRANS only applies to data blocks subsequently allocated, while a new setting for MAXTRANS applies to all blocks (already and subsequently allocated blocks) of an index.

The storage parameters INITIAL and MINEXTENTS cannot be altered. All new settings for the other storage parameters affect only extents subsequently allocated for the index.

For indexes that implement integrity constraints, you can also adjust storage parameters by issuing an ALTER TABLE statement that includes the ENABLE clause with the USING INDEX option. For example, the following statement changes the storage options of the index defined in the previous section:

```
ALTER TABLE emp
   ENABLE PRIMARY KEY USING INDEX
   PCTFREE 5;
```

Monitoring Space Use of Indexes

If key values in an index are inserted, updated, and deleted frequently, the index may or may not use its acquired space efficiently over time. Monitor an index's efficiency of space usage at regular intervals by first analyzing the index's structure and then querying the INDEX_STATS view:

```
SELECT pct_used FROM sys.index_stats WHERE name = 'indexname';
```

The percentage of an index's space usage will vary according to how often index keys are inserted, updated, or deleted. Develop a history of an index's average efficiency of space usage by performing the following sequence of operations several times: validating the index, checking PCT_USED, and dropping and re–creating the index. When you find that an index's space usage drops below its average, you can condense the index's space by dropping the index and re–creating or re–building it.

See Also: For information about analyzing an index's structure, see "Analyzing Tables, Indexes, and Clusters" on page 16 – 3.

Dropping Indexes

To drop an index, the index must be contained in your schema, or you must have the DROP ANY INDEX system privilege.

You might want to drop an index for any of the following reasons:

- The index is no longer required.
- The index is not providing anticipated performance improvements for queries issued against the associated table. (For example, the table might be very small, or there might be many rows in the table but very few index entries.)
- Applications do not use the index to query the data.
- The index has become invalid and must be dropped before being rebuilt.
- The index has become too fragmented and must be dropped before being rebuilt.

When you drop an index, all extents of the index's segment are returned to the containing tablespace and become available for other objects in the tablespace.

How you drop an index depends on whether you created the index explicitly with a CREATE INDEX statement, or implicitly by defining a key constraint on a table.

Note: If a table is dropped, all associated indexes are dropped automatically.

You cannot drop only the index associated with an enabled UNIQUE key or PRIMARY KEY constraint. To drop a constraint's associated index, you must disable or drop the constraint itself.

You can drop an explicitly created index with the SQL command DROP INDEX. For example, to drop the EMP_ENAME index, you would enter the following statement:

```
DROP INDEX emp_ename;
```

See Also: For information about analyzing indexes, see "Analyzing Tables, Indexes, and Clusters" on page 16 – 3.

For more information about dropping a constraint's associated index, see "Managing Integrity Constraints" on page 16 – 11.

CHAPTER

14

Managing Clusters

This chapter describes aspects of managing clusters (including clustered tables and indexes), and includes the following topics:

- Guidelines for Managing Clusters
- Creating Clusters
- Altering Clusters
- Dropping Clusters

Before attempting tasks described in this chapter, familiarize yourself with the concepts in Chapter 10, “Guidelines for Managing Schema Objects.”

Guidelines for Managing Clusters

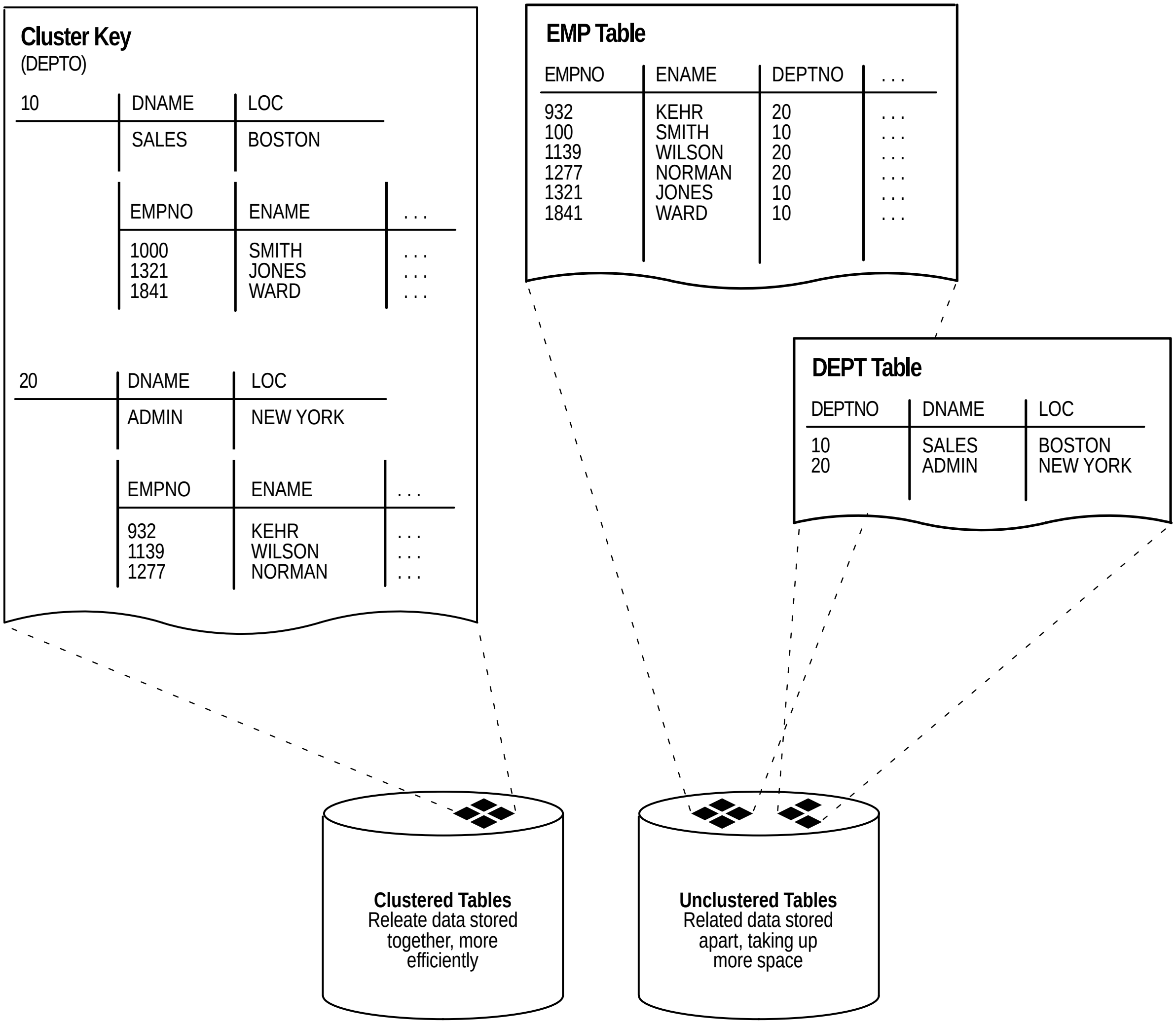
A *cluster* provides an optional method of storing table data. A cluster is comprised of a group of tables that share the same data blocks, which are grouped together because they share common columns and are often used together. For example, the EMP and DEPT table share the DEPTNO column. When you cluster the EMP and DEPT tables (see Figure 14 – 1), Oracle physically stores all rows for each department from both the EMP and DEPT tables in the same data blocks. You should not use clusters for tables that are frequently accessed individually.

Because clusters store related rows of different tables together in the same data blocks, properly used clusters offer two primary benefits:

- Disk I/O is reduced and access time improves for joins of clustered tables.
- The *cluster key* is the column, or group of columns, that the clustered tables have in common. You specify the columns of the cluster key when creating the cluster. You subsequently specify the same columns when creating every table added to the cluster. Each cluster key value is stored only once each in the cluster and the cluster index, no matter how many rows of different tables contain the value.

  Therefore, less storage might be required to store related table and index data in a cluster than is necessary in non–clustered table format. For example, notice how each cluster key (each DEPTNO) is stored just once for many rows that contain the same value in both the EMP and DEPT tables.

After creating a cluster, you can create tables in the cluster. However, before any rows can be inserted into the clustered tables, a cluster index must be created. Using clusters does not affect the creation of additional indexes on the clustered tables; they can be created and dropped as usual.



10	DNAME	LOC
	SALES	BOSTON

EMPNO	ENAME	. . .
1000	SMITH	. . .
1321	JONES	. . .
1841	WARD	. . .

20	DNAME	LOC
	ADMIN	NEW YORK

EMPNO	ENAME	. . .
932	KEHR	. . .
1139	WILSON	. . .
1277	NORMAN	. . .

EMPNO	ENAME	DEPTNO	. . .
932	KEHR	20	. . .
100	SMITH	10	. . .
1139	WILSON	20	. . .
1277	NORMAN	20	. . .
1321	JONES	10	. . .
1841	WARD	10	. . .

DEPTNO	DNAME	LOC
10	SALES	BOSTON
20	ADMIN	NEW YORK

Clustered Tables
Releate data stored together, more efficiently
Unclustered Tables
Related data stored apart, taking up more space

Figure 14 – 1 Clustered Table Data

The following sections describe guidelines to consider when managing clusters, and includes the following topics:

- Cluster Appropriate Tables
- Choose Appropriate Columns for the Cluster Key
- Specify Data Block Space Use
- Specify the Space Required by an Average Cluster Key and Its Associated Rows

- Specify the Location of Each Cluster and Cluster Index
- Estimate Cluster Size and Set Storage Parameters

See Also: For more information about clusters, see the *Oracle7 Server Concepts* manual.

Cluster Appropriate Tables

Use clusters to store one or more tables that are primarily queried (not predominantly inserted into or updated) and for which the queries often join data of multiple tables in the cluster or retrieve related data from a single table.

Choose Appropriate Columns for the Cluster Key

Choose cluster key columns carefully. If multiple columns are used in queries that join the tables, make the cluster key a composite key. In general, the characteristics that indicate a good cluster index are the same as those for any index.

A good cluster key has enough unique values so that the group of rows corresponding to each key value fills approximately one data block. Having too few rows per cluster key value can waste space and result in negligible performance gains. Cluster keys that are so specific that only a few rows share a common value can cause wasted space in blocks, unless a small SIZE was specified at cluster creation time (see below).

Too many rows per cluster key value can cause extra searching to find rows for that key. Cluster keys on values that are too general (for example, MALE and FEMALE) result in excessive searching and can result in worse performance than with no clustering.

A cluster index cannot be unique or include a column defined as LONG.

See Also: For information about characteristics of a good index, see "Guidelines for Managing Indexes" on page 13 – 2.

Specify Data Block Space Use

By specifying the PCTFREE and PCTUSED parameters during the creation of a cluster, you can affect the space utilization and amount of space reserved for updates to the current rows in the data blocks of a cluster's data segment. Note that PCTFREE and PCTUSED parameters set for tables created in a cluster are ignored; clustered tables automatically use the settings set for the cluster.

See Also: For more information about setting PCTFREE and PCTUSED, see "Managing the Space Usage of Data Blocks" on page 10 – 2 .

Specify the Space Required by an Average Cluster Key and Its Associated Rows

The CREATE CLUSTER command has an optional argument, SIZE, which is the estimated number of bytes required by an average cluster key and its associated rows. Oracle uses the SIZE parameter when performing the following tasks:

- estimating the number of cluster keys (and associated rows) that can fit in a clustered data block
- limiting the number of cluster keys placed in a clustered data block; this maximizes the storage efficiency of keys within a cluster

SIZE does not limit the space that can be used by a given cluster key. For example, if SIZE is set such that two cluster keys can fit in one data block, any amount of the available data block space can still be used by either of the cluster keys.

By default, Oracle stores only one cluster key and its associated rows in each data block of the cluster's data segment. Although block size can vary from one operating system to the next, the rule of one key per block is maintained as clustered tables are imported to other databases on other machines.

If all the rows for a given cluster key value cannot fit in one block, the blocks are chained together to speed access to all the values with the given key. The cluster index points to the beginning of the chain of blocks, each of which contains the cluster key value and associated rows. If the cluster SIZE is such that more than one key fits in a block, blocks can belong to more than one chain.

Specify the Location of Each Cluster and Cluster Index

If you have the proper privileges and tablespace quota, you can create a new cluster and the associated cluster index in any tablespace that is currently online. Always specify the TABLESPACE option in a CREATE CLUSTER/INDEX statement to identify the tablespace to store the new cluster or index.

The cluster and its cluster index can be created in different tablespaces. In fact, creating a cluster and its index in different tablespaces that are stored on different storage devices allows table data and index data to be retrieved simultaneously with minimal disk contention.

Estimate Cluster Size and Set Storage Parameters

The benefits of estimating a cluster's size before creating one follow:

- You can use the combined estimated size of clusters, along with estimates for indexes, rollback segments, and redo log files, to determine the amount of disk space that is required to hold an intended database. From these estimates, you can make correct hardware purchases and other decisions.

- You can use the estimated size of an individual cluster to better manage the disk space that the cluster will use. When a cluster is created, you can set appropriate storage parameters and improve I/O performance of applications that use the cluster.

Whether or not you estimate table size before creation, you can explicitly set storage parameters when creating each non–clustered table. Any storage parameter that you do not explicitly set when creating or subsequently altering a table automatically uses the corresponding default storage parameter set for the tablespace in which the table resides. Clustered tables also automatically use the storage parameters of the cluster.

See Also: For information about estimating the size of schema objects, including clusters, see Appendix A.

Creating Clusters

This section describes how to create clusters, and includes the following topics:

- Creating Clustered Tables
- Creating Cluster Indexes

To create a cluster in your schema, you must have the CREATE CLUSTER system privilege and a quota for the tablespace intended to contain the cluster or the UNLIMITED TABLESPACE system privilege.

To create a cluster in another user's schema, you must have the CREATE ANY CLUSTER system privilege and the owner must have a quota for the tablespace intended to contain the cluster or the UNLIMITED TABLESPACE system privilege.

You can create a cluster using the SQL command CREATE CLUSTER. The following statement creates a cluster named EMP_DEPT, which stores the EMP and DEPT tables, clustered by the DEPTNO column:

```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
   PCTUSED 80
   PCTFREE 5
   SIZE 600
   TABLESPACE users
   STORAGE (INITIAL 200k
      NEXT 300K
      MINEXTENTS 2
      MAXEXTENTS 20
      PCTINCREASE 33);
```

Creating Clustered Tables

To create a table in a cluster, you must have either the CREATE TABLE or CREATE ANY TABLE system privilege. You do not need a tablespace quota or the UNLIMITED TABLESPACE system privilege to create a table in a cluster.

You can create a table in a cluster using the SQL command CREATE TABLE with the CLUSTER option. The EMP and DEPT tables can be created in the EMP_DEPT cluster using the following statements:

```
CREATE TABLE dept (
   deptno NUMBER(3) PRIMARY KEY, . . . )
   CLUSTER emp_dept (deptno);

CREATE TABLE emp (
   empno NUMBER(5) PRIMARY KEY,
   ename VARCHAR2(15) NOT NULL,
   . . .
   deptno NUMBER(3) REFERENCES dept)
   CLUSTER emp_dept (deptno);
```

> **Note:** You can specify the schema for a clustered table in the CREATE TABLE statement; a clustered table can be in a different schema than the schema containing the cluster.

Creating Cluster Indexes

To create a cluster index, one of the following conditions must be true:

- Your schema contains the cluster and you have the CREATE INDEX system privilege.
- You have the CREATE ANY INDEX system privilege.

In either case, you must also have either a quota for the tablespace intended to contain the cluster index, or the UNLIMITED TABLESPACE system privilege.

A cluster index must be created before any rows can be inserted into any clustered table. The following statement creates a cluster index for the EMP_DEPT cluster:

```
CREATE INDEX emp_dept_index
   ON CLUSTER emp_dept
   INITRANS 2
   MAXTRANS 5
   TABLESPACE users
   STORAGE (INITIAL 50K
      NEXT 50K
      MINEXTENTS 2
      MAXEXTENTS 10
      PCTINCREASE 33)
   PCTFREE 5;
```

The cluster key establishes the relationship of the tables in the cluster. Several storage settings are explicitly specified for the cluster and cluster index.

See Also: See Chapter 20 for more information about system privileges, and Chapter 19 for information about tablespace quotas.

Altering Clusters

You can alter an existing cluster to change the following settings:

- data block space usage parameters (PCTFREE, PCTUSED)
- the average cluster key size (SIZE)
- transaction entry settings (INITRANS, MAXTRANS)
- storage parameters (NEXT, PCTINCREASE)

To alter a cluster, your schema must contain the cluster or you must have the ALTER ANY CLUSTER system privilege.

When you alter data block space usage parameters (PCTFREE and PCTUSED) or the cluster size parameter (SIZE) of a cluster, the new settings apply to all data blocks used by the cluster, including blocks already allocated and blocks subsequently allocated for the cluster. Blocks already allocated for the table are reorganized when necessary (not immediately).

When you alter the transaction entry settings (INITRANS, MAXTRANS) of a cluster, a new setting for INITRANS applies only to data blocks subsequently allocated for the cluster, while a new setting for MAXTRANS applies to all blocks (already and subsequently allocated blocks) of a cluster.

The storage parameters INITIAL and MINEXTENTS cannot be altered. All new settings for the other storage parameters affect only extents subsequently allocated for the cluster.

To alter a cluster, use the SQL command ALTER CLUSTER. The following statement alters the EMP_DEPT cluster:

```
ALTER CLUSTER emp_dept
   PCTFREE 30
   PCTUSED 60;
```

Altering Cluster Tables and Cluster Indexes

You can alter clustered tables using the SQL command ALTER TABLE. However, any data block space parameters, transaction entry parameters, or storage parameters you set in an ALTER TABLE statement for a clustered table generate an error message (ORA–01771, “illegal option for a clustered table”). Oracle uses the parameters of the cluster for all clustered tables. Therefore, you can use the ALTER TABLE command only to add or modify columns, or add, drop, enable, or disable integrity constraints or triggers for a clustered table.

> **Note:** When estimating the size of cluster indexes, remember that the index is on each cluster key, not the actual rows; therefore, each key will only appear once in the index.

Manually Allocating Storage for a Cluster

Oracle dynamically allocates additional extents for the data segment of a cluster as required. In some circumstances, however, you might want to allocate an additional extent for a cluster explicitly. For example, when using the Oracle Parallel Server, you can allocate an extent of a cluster explicitly for a specific instance.

You allocate a new extent for a cluster using the SQL command ALTER CLUSTER with the ALLOCATE EXTENT option.

See Also: For information about altering tables, see “Altering Tables” on page 11 – 7.

You alter cluster indexes exactly as you do other indexes. For more information, see “Altering an Index” on page 13 – 8.

For more information about the CLUSTER parameter in the ALTER CLUSTER command, see the *Oracle7 Parallel Server Concepts & Administration* guide.

Dropping Clusters

This section describes aspects of dropping clusters, and includes the following topics:

- Dropping Clustered Tables
- Dropping Cluster Indexes

A cluster can be dropped if the tables within the cluster are no longer necessary. When a cluster is dropped, so are the tables within the cluster and the corresponding cluster index; all extents belonging to both the cluster’s data segment and the index segment of the cluster index are returned to the containing tablespace and become available for other segments within the tablespace.

Dropping Clustered Tables

To drop a cluster, your schema must contain the cluster or you must have the DROP ANY CLUSTER system privilege. You do not have to have additional privileges to drop a cluster that contains tables, even if the clustered tables are not owned by the owner of the cluster.

Clustered tables can be dropped individually without affecting the table's cluster, other clustered tables, or the cluster index. A clustered table is dropped just as a non–clustered table is dropped—with the SQL command DROP TABLE.

> **Note:** When you drop a single table from a cluster, Oracle deletes each row of the table individually. To maximize efficiency when you intend to drop an entire cluster, drop the cluster including all tables by using the DROP CLUSTER command with the INCLUDING TABLES option. Drop an individual table from a cluster (using the DROP TABLE command) only if you want the rest of the cluster to remain.

See Also: For information about dropping a table, see "Dropping Tables" on page 11 – 9.

Dropping Cluster Indexes

A cluster index can be dropped without affecting the cluster or its clustered tables. However, clustered tables cannot be used if there is no cluster index; you must re–create the cluster index to allow access to the cluster. Cluster indexes are sometimes dropped as part of the procedure to rebuild a fragmented cluster index.

To drop a cluster that contains no tables, and its cluster index, use the SQL command DROP CLUSTER. For example, the following statement drops the empty cluster named EMP_DEPT:

```
DROP CLUSTER emp_dept;
```

If the cluster contains one or more clustered tables and you intend to drop the tables as well, add the INCLUDING TABLES option of the DROP CLUSTER command, as follows:

```
DROP CLUSTER emp_dept INCLUDING TABLES;
```

If the INCLUDING TABLES option is not included and the cluster contains tables, an error is returned.

If one or more tables in a cluster contain primary or unique keys that are referenced by FOREIGN KEY constraints of tables outside the cluster, the cluster cannot be dropped unless the dependent FOREIGN KEY constraints are also dropped. This can be easily done using the CASCADE CONSTRAINTS option of the DROP CLUSTER command, as shown in the following example:

```
DROP CLUSTER emp_dept INCLUDING TABLES CASCADE CONSTRAINTS;
```

Oracle returns an error if you do not use the CASCADE CONSTRAINTS option and constraints exist.

See Also: For information about dropping an index, see “Dropping Indexes” on page 13 – 9.

CHAPTER

15 Managing Hash Clusters

This chapter describes how to manage hash clusters, and includes the following topics:

- Guidelines for Managing Hash Clusters
- Creating Hash Clusters
- Altering Hash Clusters
- Dropping Hash Clusters

See Also: Before attempting tasks described in this chapter, familiarize yourself with the concepts in Chapter 10, "Guidelines for Managing Schema Objects."

Guidelines for Managing Hash Clusters

This section describes guidelines to consider before attempting to manage hash clusters, and includes the following topics:

- Advantages of Hashing
- Disadvantages of Hashing
- Estimate Size Required by Hash Clusters and Set Storage Parameters

Storing a table in a hash cluster is an optional way to improve the performance of data retrieval. A hash cluster provides an alternative to a non–clustered table with an index or an index cluster. With an indexed table or index cluster, Oracle locates the rows in a table using key values that Oracle stores in a separate index. To use hashing, you create a hash cluster and load tables into it. Oracle physically stores the rows of a table in a hash cluster and retrieves them according to the results of a hash function.

Oracle uses a *hash function* to generate a distribution of numeric values, called *hash values*, which are based on specific cluster key values. The key of a hash cluster, like the key of an index cluster, can be a single column or composite key (multiple column key). To find or store a row in a hash cluster, Oracle applies the hash function to the row's cluster key value; the resulting hash value corresponds to a data block in the cluster, which Oracle then reads or writes on behalf of the issued statement.

To find or store a row in an indexed table or cluster, a minimum of two (there are usually more) I/Os must be performed:

- one or more I/Os to find or store the key value in the index
- another I/O to read or write the row in the table or cluster

In contrast, Oracle uses a hash function to locate a row in a hash cluster; no I/O is required. As a result, a minimum of one I/O operation is necessary to read or write a row in a hash cluster.

Advantages of Hashing

If you opt to use indexing rather than hashing, consider whether to store a table individually or as part of a cluster.

Hashing is most advantageous when you have the following conditions:

- Most queries are equality queries on the cluster key:

```
SELECT . . . WHERE cluster_key = . . . ;
```

In such cases, the cluster key in the equality condition is hashed, and the corresponding hash key is usually found with a single read. In comparison, for an indexed table the key value must first be found in the index (usually several reads), and then the row is read from the table (another read).

- The tables in the hash cluster are primarily static in size so that you can determine the number of rows and amount of space required for the tables in the cluster. If tables in a hash cluster require more space than the initial allocation for the cluster, performance degradation can be substantial because overflow blocks are required.

Disadvantages of Hashing

Hashing is not advantageous in the following situations:

- Most queries on the table retrieve rows over a range of cluster key values. For example, in full table scans, or queries like the following, a hash function cannot be used to determine the location of specific hash keys; instead, the equivalent of a full table scan must be done to fetch the rows for the query:

  ```
  SELECT . . . WHERE cluster_key < . . . ;
  ```

  With an index, key values are ordered in the index, so cluster key values that satisfy the WHERE clause of a query can be found with relatively few I/Os.

- The table is not static and continually growing. If a table grows without limit, the space required over the life of the table (its cluster) cannot be pre–determined.

- Applications frequently perform full–table scans on the table and the table is sparsely populated. A full–table scan in this situation takes longer under hashing.

- You cannot afford to pre–allocate the space that the hash cluster will eventually need.

See Also: For more information about creating hash clusters and specifying hash functions see the *Oracle7 Server SQL Reference.*

For information about hash functions and specifying user–defined hash functions, see the *Oracle7 Server Concepts* manual.

Even if you decide to use hashing, a table can still have separate indexes on any columns, including the cluster key. See the *Oracle7 Server Application Developer's Guide* for additional recommendations.

Estimate Size Required by Hash Clusters and Set Storage Parameters

As with index clusters, it is important to estimate the storage required for the data in a hash cluster.

Oracle guarantees that the initial allocation of space is sufficient to store the hash table according to the settings SIZE and HASHKEYS. If settings for the storage parameters INITIAL, NEXT, and MINEXTENTS do not account for the hash table size, incremental (additional) extents are allocated until at least SIZE*HASHKEYS is reached. For example, assume that the data block size is 2K, the available data space per block is approximately 1900 bytes (data block size minus overhead), and that the STORAGE and HASH parameters are specified in the CREATE CLUSTER command as follows:

```
STORAGE (INITIAL 100K
   NEXT 150K
   MINEXTENTS 1
   PCTINCREASE 0)
SIZE 1500
HASHKEYS 100
```

In this example, only one hash key can be assigned per data block. Therefore, the initial space required for the hash cluster is at least 100*2K or 200K. The settings for the storage parameters do not account for this requirement. Therefore, an initial extent of 100K and a second extent of 150K are allocated to the hash cluster.

Alternatively, assume the HASH parameters are specified as follows:

```
SIZE 500 HASHKEYS 100
```

In this case, three hash keys are assigned to each data block. Therefore, the initial space required for the hash cluster is at least 34*2K or 68K. The initial settings for the storage parameters are sufficient for this requirement (an initial extent of 100K is allocated to the hash cluster).

See Also: To estimate the size of a hash cluster, use the procedure given in “Estimating Space Required by Clusters” on page A – 1, along with the supplemental information in “Estimating Space Required by Hash Clusters” on page A – 14.

Creating Hash Clusters

After a hash cluster is created, tables can be created in the cluster. A hash cluster is created using the SQL command CREATE CLUSTER. For example, the following statement creates a cluster named TRIAL_CLUSTER that stores the TRIAL table, clustered by the TRIALNO column:

```
CREATE CLUSTER trial_cluster (trialno NUMBER(5,0))
   PCTUSED 80
   PCTFREE 5
   TABLESPACE users
   STORAGE (INITIAL 250K     NEXT 50K
      MINEXTENTS 1     MAXEXTENTS 3
      PCTINCREASE 0)
   SIZE 2K
   HASH IS trialno HASHKEYS 150;

CREATE TABLE trial (
   trialno         NUMBER(5,0) PRIMARY KEY,
   ...)
   CLUSTER trial_cluster (trialno);
```

The following sections explain setting the parameters of the CREATE CLUSTER command specific to hash clusters.

See Also: For additional information about creating tables in a cluster, guidelines for setting other parameters of the CREATE CLUSTER command, and the privileges required to create a hash cluster, see "Creating Clusters" on page 14 – 6.

Controlling Space Use Within a Hash Cluster

When creating a hash cluster, it is important to choose the cluster key correctly and set the HASH IS, SIZE, and HASHKEYS parameters so that performance and space use are optimal. The following guidelines describe how to set these parameters.

Choosing the Key

Choosing the correct cluster key is dependent on the most common types of queries issued against the clustered tables. For example, consider the EMP table in a hash cluster. If queries often select rows by employee number, the EMPNO column should be the cluster key; if queries often select rows by department number, the DEPTNO column should be the cluster key. For hash clusters that contain a single table, the cluster key is typically the entire primary key of the contained table.

The key of a hash cluster, like that of an index cluster, can be a single column or a composite key (multiple column key). A hash cluster with a composite key must use Oracle's internal hash function.

Setting HASH IS

Only specify the HASH IS parameter if the cluster key is a single column of the NUMBER datatype, and contains uniformly distributed integers. If the above conditions apply, you can distribute rows in the cluster so that each unique cluster key value hashes, with no collisions, to a unique hash value. If these conditions do not apply, omit this option so that you use the internal hash function.

Setting SIZE

SIZE should be set to the average amount of space required to hold all rows for any given hash key. Therefore, to properly determine SIZE, you must be aware of the characteristics of your data:

- If the hash cluster is to contain only a single table and the hash key values of the rows in that table are unique (one row per value), SIZE can be set to the average row size in the cluster.
- If the hash cluster is to contain multiple tables, SIZE can be set to the average amount of space required to hold all rows associated with a representative hash value.

See Also: To estimate a preliminary value for SIZE, follow the procedures given in "Estimating Space Required by Hash Clusters" on page A – 14. If the preliminary value for SIZE is small (more than four hash keys can be assigned per data block), you can use this value for SIZE in the CREATE CLUSTER command.

However, if the value of SIZE is large (fewer than five hash keys can be assigned per data block), you should also consider the expected frequency of collisions and whether performance of data retrieval or efficiency of space usage is more important to you:

- If the hash cluster does not use the internal hash function (if you specified HASH IS) and you expect little or no collisions, you can set SIZE as estimated; no collisions occur and space is used as efficiently as possible.
- If you expect frequent collisions on inserts, the likelihood of overflow blocks being allocated to store rows is high. To reduce the possibility of overflow blocks and maximize performance when collisions are frequent, you should increase SIZE according to Table 15 – 1.

Available Space per Block/Calc'd SIZE	Setting for SIZE
1	Calculated SIZE
2	Calculated SIZE + 15%
3	Calculated SIZE + 12%
4	Calculated SIZE + 8%
>4	Calculated SIZE

Table 15 – 1 SIZE Increase Chart

Overestimating the value of SIZE increases the amount of unused space in the cluster. If space efficiency is more important than the performance of data retrieval, disregard the above adjustments and use the estimated value for SIZE.

Setting HASHKEYS

For maximum distribution of rows in a hash cluster, HASHKEYS should always be a prime number.

For example, suppose you cluster the EMP table by DEPTNO, and there are 100 DEPTNOs, with values 10, 20, . . ., 1000. Assuming you bypass the internal hash function and you create a cluster with HASHKEYS of 100, then department 10 will hash to 10, department 20 to 20, . . ., department 110 to 10 (110 mod 100), department 120 to 20, and so on. Notice that there are 10 entries for hash values of 10, 20, . . ., but none for 1, 2, . . ., and so on. As a result, there is a lot of wasted space and possibly a lot of overflow blocks because of collisions. Alternatively, if HASHKEYS is set to 101, then each department number hashes to a unique hash key value.

Controlling Space in Hash Clusters: Examples

The following examples show how to correctly choose the cluster key and set the HASH IS, SIZE, and HASHKEYS parameters. For all examples, assume that the data block size is 2K and that on average, 1950 bytes of each block is available data space (block size minus overhead).

Example 1

You decide to load the EMP table into a hash cluster. Most queries retrieve employee records by their employee number. You estimate that the maximum number of rows in the EMP table at any given time is 10000 and that the average row size is 55 bytes.

In this case, EMPNO should be the cluster key. Since this column contains integers that are unique, the internal hash function can be bypassed. SIZE can be set to the average row size, 55 bytes; note that 34 hash keys are assigned per data block. HASHKEYS can be set to the number of rows in the table, 10000, rounded up to the next highest prime number, 10001:

```
CREATE CLUSTER emp_cluster (empno NUMBER)
. . .
SIZE 55
HASH IS empno HASHKEYS 10001;
```

Example 2 Conditions similar to the previous example exist. In this case, however, rows are usually retrieved by department number. At most, there are 1000 departments with an average of 10 employees per department. Note that department numbers increment by 10 (0, 10, 20, 30, . . .).

In this case, DEPTNO should be the cluster key. Since this column contains integers that are uniformly distributed, the internal hash function can be bypassed. A pre–estimated SIZE (the average amount of space required to hold all rows per department) is 55 bytes * 10, or 550 bytes. Using this value for SIZE, only three hash keys can be assigned per data block. If you expect some collisions and want maximum performance of data retrieval, slightly alter your estimated SIZE to prevent collisions from requiring overflow blocks. By adjusting SIZE by 12%, to 620 bytes (see previous section about setting SIZE for clarification), only three hash keys are assigned per data block, leaving more space for rows from expected collisions.

HASHKEYS can be set to the number of unique department numbers, 1000, rounded up to the next highest prime number, 1009:

```
CREATE CLUSTER emp_cluster (deptno NUMBER)
. . .
SIZE 620
HASH IS deptno HASHKEYS 1009;
```

Altering Hash Clusters

You can alter a hash cluster with the SQL command ALTER CLUSTER:

```
ALTER CLUSTER emp_dept . . . ;
```

The implications for altering a hash cluster are identical to those for altering an index cluster. However, note that the SIZE, HASHKEYS, and HASH IS parameters cannot be specified in an ALTER CLUSTER statement. You must re–create the cluster to change these parameters and then copy the data from the original cluster.

See Also: For more information about altering an index cluster, see “Altering Clusters” on page 14 – 8.

Dropping Hash Clusters

You can drop a hash cluster using the SQL command DROP CLUSTER:

```
DROP CLUSTER emp_dept;
```

A table in a hash cluster is dropped using the SQL command DROP TABLE. The implications of dropping hash clusters and tables in hash clusters are the same for index clusters.

See Also: For more information about dropping clusters, see "Dropping Clusters" on page 14 – 9.

CHAPTER

16

General Management of Schema Objects

This chapter describes general schema object management issues that fall outside the scope of chapters 10 through 15, and includes the following topics:

- Creating Multiple Tables and Views in a Single Operation
- Renaming Schema Objects
- Analyzing Tables, Indexes, and Clusters
- Truncating Tables and Clusters
- Enabling and Disabling Triggers
- Managing Integrity Constraints
- Managing Object Dependencies
- Managing Object Name Resolution
- Changing Storage Parameters for the Data Dictionary
- Displaying Information About Schema Objects

Creating Multiple Tables and Views in A Single Operation

To create schema objects you must have the required privileges for any included operation. For example, to create multiple tables using the CREATE SCHEMA command, you must have the privileges required to create tables.

You can create several tables and views and grant privileges in one operation using the SQL command CREATE SCHEMA. The CREATE SCHEMA command is useful if you want to guarantee the creation of several tables and views and grants in one operation. If an individual table, view or grant fails, the entire statement is rolled back. None of the objects are created, nor are the privileges granted. The following statement creates two tables and a view that joins data from the two tables:

```
CREATE SCHEMA AUTHORIZATION scott
   CREATE TABLE dept (
      deptno     NUMBER(3,0) PRIMARY KEY,
      dname      VARCHAR2(15),
      loc        VARCHAR2(25)
   CREATE TABLE emp (
     empno       NUMBER(5,0) PRIMARY KEY,
      ename      VARCHAR2(15) NOT NULL,
      job        VARCHAR2(10),
      mgr        NUMBER(5,0),
      hiredate   DATE DEFAULT (sysdate),
      sal        NUMBER(7,2),
      comm       NUMBER(7,2),
      deptno     NUMBER(3,0) NOT NULL
      CONSTRAINT dept_fkey REFERENCES dept)
   CREATE VIEW sales_staff AS
      SELECT empno, ename, sal, comm
         FROM emp
      WHERE deptno = 30
      WITH CHECK OPTION CONSTRAINT sales_staff_cnst
   GRANT SELECT ON sales_staff TO human_resources;
```

The CREATE SCHEMA command does not support Oracle extensions to the ANSI CREATE TABLE and CREATE VIEW commands; this includes the STORAGE clause.

Renaming Schema Objects

To rename an object, you must own it. You can rename schema objects in either of the following ways:

- drop and re–create the object
- rename the object using the SQL command RENAME

If you drop and re–create an object, all privilege grants for that object are lost. Privileges must be re–granted when the object is re–created. Alternatively, a table, view, sequence, or a private synonym of a table, view, or sequence can be renamed using the RENAME command. When using the RENAME command, grants made for the object are carried forward for the new name. For example, the following statement renames the SALES_STAFF view:

```
RENAME sales_staff TO dept_30;
```

> **Note:** You cannot rename a stored PL/SQL program unit, public synonym, index, or cluster. To rename such an object, you must drop and re–create it.

Before renaming a schema object, consider the following effects:

- All views and PL/SQL program units dependent on a renamed object become invalid, and must be recompiled before next use.
- All synonyms for a renamed object return an error when used.

See Also: For more information about how Oracle manages object dependencies, see page 16 – 18.

Analyzing Tables, Indexes, and Clusters

This section describes how to analyze tables, indexes, and clusters, and includes the following topics:

- Using Statistics for Tables, Indexes, and Clusters
- Validating Tables, Indexes, and Clusters
- Listing Chained Rows of Tables and Clusters

You can analyze a table, index, or cluster to gather data about it, or to verify the validity of its storage format. To analyze a table, cluster, or index, you must own the table, cluster, or index or have the ANALYZE ANY system privilege.

These schema objects can also be analyzed to collect or update statistics about specific objects. When a DML statement is issued, the statistics for the referenced objects are used to determine the most efficient execution plan for the statement. This optimization is called “cost–based optimization.“ The statistics are stored in the data dictionary.

A table, index, or cluster can be analyzed to *validate* the structure of the object. For example, in rare cases such as hardware or other system failures, an index can become corrupted and not perform correctly. When validating the index, you can confirm that every entry in the index points to the correct row of the associated table. If a schema object is corrupt, you can drop and re–create it.

A table or cluster can be analyzed to collect information about chained rows of the table or cluster. These results are useful in determining whether you have enough room for updates to rows. For example, this information can show whether PCTFREE is set appropriately for the table or cluster.

See Also: For more information about analyzing tables, indexes, and clusters for performance statistics and the optimizer, see the *Oracle7 Server Tuning* guide.

Using Statistics for Tables, Indexes, and Clusters

Statistics about the physical storage characteristics of a table, index, or cluster can be gathered and stored in the data dictionary using the SQL command ANALYZE with the STATISTICS option. Oracle can use these statistics when cost–based optimization is employed to choose the most efficient execution plan for SQL statements accessing analyzed objects. You can also use statistics generated by this command to write efficient SQL statements that access analyzed objects.

You can compute or estimate statistics using the ANALYZE command, with either the COMPUTE STATISTICS or ESTIMATE STATISTICS option:

COMPUTE STATISTICS

When computing statistics, an entire object is scanned to gather data about the object. This data is used by Oracle to compute exact statistics about the object. Slight variances throughout the object are accounted for in these computed statistics. Because an entire object is scanned to gather information for computed statistics, the larger the size of an object, the more work that is required to gather the necessary information.

ESTIMATE STATISTICS

When estimating statistics, Oracle gathers representative information from portions of an object. This subset of information provides reasonable, estimated statistics about the object. The accuracy of estimated statistics depends upon how representative the sampling used by Oracle is. Only parts of an object are scanned to gather information for estimated statistics, so an object can be analyzed quickly. You can optionally specify the number or percentage of rows that Oracle should use in making the estimate.

Note: When calculating statistics for tables or clusters, the amount of temporary space required to perform the calculation is related to the number of rows specified. For COMPUTE STATISTICS, enough temporary space to hold and sort the entire table plus a small overhead for each row is required. For ESTIMATE STATISTICS, enough temporary space to hold and sort the requested sample of rows plus a small overhead for each row is required. For indexes, no temporary space is required for analyzing.

Viewing Object Statistics

Whether statistics for an object are computed or estimated, the statistics are stored in the data dictionary. The statistics can be queried using the following data dictionary views:

- USER_INDEXES, ALL_INDEXES, DBA_INDEXES
- USER_TABLES, ALL_TABLES, DBA_TABLES
- USER_TAB_COLUMNS, ALL_TAB_COLUMNS, DBA_TAB_COLUMNS

Note: Rows in these views contain entries in the statistics columns only for indexes, tables, and clusters for which you have gathered statistics. The entries are updated for an object each time you ANALYZE the object.

Table Statistics You can gather the following statistics on a table:

Note: The * symbol indicates that the numbers will always be an exact value when computing statistics.

- number of rows
- number of blocks that have been used *
- number of blocks never used
- average available free space
- number of chained rows
- average row length

- number of distinct values per column
- the second smallest value per column *
- the second largest value per column *

> **Note:** Statistics for all indexes associated with a table are automatically gathered when the table is analyzed.

Index Statistics You can gather the following statistics on an index:

- index level *
- number of leaf blocks
- number of distinct keys
- average number of leaf blocks / key
- average number of data blocks / key
- clustering factor
- minimum key value *
- maximum key value*

Cluster Statistics The only statistic that can be gathered for a cluster is the average cluster key chain length; this statistic can be estimated or computed. Statistics for tables in a cluster and all indexes associated with the cluster's tables (including the cluster key index) are automatically gathered when the cluster is analyzed for statistics.

> **Note:** If the data dictionary currently contains statistics for the specified object when an ANALYZE statement is issued, the new statistics replace the old statistics in the data dictionary.

Computing Statistics

The following statement computes statistics for the EMP table:

```
ANALYZE TABLE emp COMPUTE STATISTICS;
```

The following query estimates statistics on the EMP table, using the default statistical sample of 1064 rows:

```
ANALYZE TABLE emp ESTIMATE STATISTICS;
```

To specify the statistical sample that Oracle should use, include the SAMPLE option with the ESTIMATE STATISTICS option. You can specify an integer that indicates either a number of rows or index values, or a percentage of the rows or index values in the table. The following statements show examples of each option:

```
ANALYZE TABLE emp
   ESTIMATE STATISTICS
      SAMPLE 2000 ROWS;
ANALYZE TABLE emp
   ESTIMATE STATISTICS
      SAMPLE 33 PERCENT;
```

In either case, if you specify a percentage greater than 50, or a number of rows or index values that is greater than 50% of those in the object, Oracle computes the exact statistics, rather than estimating.

Removing Statistics for a Schema Object

You can remove statistics for a table, index, or cluster from the data dictionary using the ANALYZE command with the DELETE STATISTICS option. For example, you might want to delete statistics for an object if you do not want cost–based optimization to be used for statements regarding the object. The following statement deletes statistics for the EMP table from the data dictionary:

```
ANALYZE TABLE emp DELETE STATISTICS;
```

Shared SQL and Analyzing Statistics

Analyzing a table, cluster, or index can affect current shared SQL statements, which are statements currently in the shared pool. Whenever an object is analyzed to update or delete statistics, all shared SQL statements that reference the analyzed object are flushed from memory so that the next execution of the statement can take advantage of the new statistics.

You can call the following procedures:

DBMS_UTILITY.–ANALYZE_SCHEMA()
This procedure takes two arguments, the name of a schema and an analysis method ('COMPUTE', 'ESTIMATE', or 'DELETE'), and gathers statistics on all of the objects in the schema.

DBMS_DDL.–ANALYZE_OBJECT()
This procedure takes four arguments, the type of an object ('CLUSTER', 'TABLE', or 'INDEX'), the schema of the object, the name of the object, and an analysis method ('COMPUTE', 'ESTIMATE', or 'DELETE'), and gathers statistics on the object.

You should call these procedures periodically to update the statistics.

Validating Tables, Indexes, and Clusters

To verify the integrity of the structure of a table, index, cluster, or snapshot, use the ANALYZE command with the VALIDATE STRUCTURE option. If the structure is valid, no error is returned. However, if the structure is corrupt, you receive an error message. If a table, index, or cluster is corrupt, you should drop it and re–create it. If a snapshot is corrupt, perform a complete refresh and ensure that you have remedied the problem; if not, drop and re–create the snapshot.

The following statement analyzes the EMP table:

```
ANALYZE TABLE emp VALIDATE STRUCTURE;
```

You can validate an object and all related objects by including the CASCADE option. The following statement validates the EMP table and all associated indexes:

```
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE;
```

Listing Chained Rows of Tables and Clusters

You can look at the chained and migrated rows of a table or cluster using the ANALYZE command with the LIST CHAINED ROWS option. The results of this command are stored in a specified table created explicitly to accept the information returned by the LIST CHAINED ROWS option.

To create an appropriate table to accept data returned by an ANALYZE... LIST CHAINED ROWS statement, use the UTLCHAIN.SQL script provided with Oracle. The UTLCHAIN.SQL script creates a table named CHAINED_ROWS in the schema of the user submitting the script.

After a CHAINED_ROWS table is created, you can specify it when using the ANALYZE command. For example, the following statement inserts rows containing information about the chained rows in the EMP_DEPT cluster into the CHAINED_ROWS table:

```
ANALYZE CLUSTER emp_dept LIST CHAINED ROWS INTO chained_rows;
```

See Also: The name and location of the UTLCHAIN.SQL script are operating system–dependent; see your operating system–specific Oracle documentation.

For more information about reducing the number of chained and migrated rows in a table or cluster, see *Oracle7 Server Tuning*.

Truncating Tables and Clusters

You can delete all rows of a table or all rows in a group of clustered tables so that the table (or cluster) still exists, but is completely empty. For example, you may have a table that contains monthly data, and at the end of each month, you need to empty it (delete all rows) after archiving its data.

To delete all rows from a table, you have three options:

1. Using the DELETE command

   You can delete the rows of a table using the DELETE command. For example, the following statement deletes all rows from the EMP table:

   ```
   DELETE FROM emp;
   ```

2. Using the DROP and CREATE commands

   You can drop a table and then re–create the table. For example, the following statements drop and then re–create the EMP table:

   ```
   DROP TABLE emp;
   CREATE TABLE emp ( . . . );
   ```

3. Using TRUNCATE

   You can delete all rows of the table using the SQL command TRUNCATE. For example, the following statement truncates the EMP table:

   ```
   TRUNCATE TABLE emp;
   ```

Using DELETE

If there are many rows present in a table or cluster when using the DELETE command, significant system resources are consumed as the rows are deleted. For example, CPU time, redo log space, and rollback segment space from the table and any associated indexes require resources. Also, as each row is deleted, triggers can be fired. The space previously allocated to the resulting empty table or cluster remains associated with that object.

Using DROP and CREATE

When dropping and re–creating a table or cluster, all associated indexes, integrity constraints, and triggers are also dropped, and all objects that depend on the dropped table or clustered table are invalidated. Also, all grants for the dropped table or clustered table are dropped.

Using TRUNCATE

Using the TRUNCATE command provides a fast, efficient method for deleting all rows from a table or cluster. A TRUNCATE statement does not generate any rollback information and it commits immediately; it is a DDL statement and cannot be rolled back. A TRUNCATE statement does not affect any structures associated with the table being truncated (constraints and triggers) or authorizations. A TRUNCATE statement also specifies whether space currently allocated for the table is returned to the containing tablespace after truncation.

You can truncate any table or cluster in the user's associated schema. Also, any user that has the DELETE ANY TABLE system privilege can truncate a table or cluster in any schema.

Before truncating a table or clustered table containing a parent key, all referencing foreign keys in different tables must be disabled. A self–referential constraint does not have to be disabled.

As a TRUNCATE statement deletes rows from a table, triggers associated with the table are not fired. Also, a TRUNCATE statement does not generate any audit information corresponding to DELETE statements if auditing is enabled. Instead, a single audit record is generated for the TRUNCATE statement being issued.

A hash cluster cannot be truncated. Also, tables within a hash or index cluster cannot be individually truncated; truncation of an index cluster deletes all rows from all tables in the cluster. If all the rows must be deleted from an individual clustered table, use the DELETE command or drop and re–create the table.

The REUSE STORAGE or DROP STORAGE options of the TRUNCATE command control whether space currently allocated for a table or cluster is returned to the containing tablespace after truncation. The default option, DROP STORAGE, reduces the number of extents allocated to the resulting table to the original setting for MINEXTENTS. Freed extents are then returned to the system and can be used by other objects.

Alternatively, the REUSE STORAGE option specifies that all space currently allocated for the table or cluster remains allocated to it. For example, the following statement truncates the EMP_DEPT cluster, leaving all extents previously allocated for the cluster available for subsequent inserts and deletes:

```
TRUNCATE CLUSTER emp_dept REUSE STORAGE;
```

The REUSE or DROP STORAGE option also applies to any associated indexes. When a table or cluster is truncated, all associated indexes are also truncated. Also note that the storage parameters for a truncated table, cluster, or associated indexes are not changed as a result of the truncation.

See Also: See Chapter 21 for information about auditing.

Enabling and Disabling Triggers

This section describes database trigger management, and includes the following topics:

- Enabling Triggers
- Disabling Triggers

Oracle enables you to define procedures, called *database triggers*, that are implicitly executed when an INSERT, UPDATE, or DELETE statement is issued against an associated table.

A trigger can be in either of two distinct modes:

enabled — An enabled trigger executes its trigger body if a triggering statement is issued and the trigger restriction, if any, evaluates to TRUE.

disabled — A disabled trigger does not execute its trigger body, even if a triggering statement is issued and the trigger restriction (if any) evaluates to TRUE.

To enable or disable triggers using the ALTER TABLE command, you must own the table, have the ALTER object privilege for the table, or have the ALTER ANY TABLE system privilege. To enable or disable an individual trigger using the ALTER TRIGGER command, you must own the trigger or have the ALTER ANY TRIGGER system privilege.

Enabling Triggers

You enable a disabled trigger using the ALTER TRIGGER command with the ENABLE option. To enable the disabled trigger named REORDER on the INVENTORY table, enter the following statement:

```
ALTER TRIGGER reorder ENABLE;
```

To enable all triggers defined for a specific table, use the ALTER TABLE command with the ENABLE clause and ALL TRIGGERS option. To enable all triggers defined for the INVENTORY table, enter the following statement:

```
ALTER TABLE inventory
   ENABLE ALL TRIGGERS;
```

Disabling Triggers

You may want to temporarily disable a trigger if one of the following conditions is true:

- An object that the trigger references is not available.
- You have to perform a large data load and want it to proceed quickly without firing triggers.
- You are loading data into the table to which the trigger applies.

By default, triggers are enabled when first created. You disable a trigger using the ALTER TRIGGER command with the DISABLE option. To disable the trigger REORDER on the INVENTORY table, enter the following statement:

```
ALTER TRIGGER reorder DISABLE;
```

You can disable all triggers associated with a table at the same time using the ALTER TABLE command with the DISABLE clause and ALL TRIGGERS option. For example, to disable all triggers defined for the INVENTORY table, enter the following statement:

```
ALTER TABLE inventory
   DISABLE ALL TRIGGERS;
```

Managing Integrity Constraints

This section explains the mechanisms and procedures for managing integrity constraints, and includes the following topics:

- Managing Constraints That Have Associated Indexes
- Enabling and Disabling Integrity Constraints Upon Definition
- Enabling and Disabling Existing Integrity Constraints
- Dropping Integrity Constraints
- Reporting Constraint Exceptions

An integrity constraint defined on a table can be in one of two modes:

enabled

When a constraint is enabled, the rule defined by the constraint is enforced on the data values in the columns that define the constraint. The definition of the constraint is stored in the data dictionary.

disabled

When a constraint is disabled, the rule defined by the constraint is not enforced on the data values in the columns included in the constraint; however, the definition of the constraint is retained in the data dictionary.

You can think of an integrity constraint as a statement about the data in a database. This statement is always not false when the constraint is enabled. However, the statement may or may not be true when the constraint is disabled because data in violation of the integrity constraint can be in the database.

To enforce the rules defined by integrity constraints, the constraints should always be enabled. In certain situations it is desirable to temporarily disable the integrity constraints of a table for the following performance reasons:

- when loading large amounts of data into a table using SQL*Loader
- when performing batch operations that make massive changes to a table (for example, changing every employee's number by adding 1000 to the existing number)
- when importing or exporting one table at a time

In all three cases, temporarily disabling integrity constraints can improve the performance of the operation.

While a constraint is enabled, no row violating the constraint can be inserted into the table. While the constraint is disabled, though, such a row can be inserted; this row is known as an *exception* to the constraint. While exceptions to a constraint can exist in a table, *the constraint cannot be enabled*. The rows that violate the constraint must be either updated or deleted in order for the constraint to be enabled.

See Also: You can identify exceptions to a specific integrity constraint while attempting to enable the constraint. See "Reporting Constraint Exceptions" on page 16 – 16.

Managing Constraints That Have Associated Indexes

An index associated with a UNIQUE key or PRIMARY KEY constraint is automatically created by Oracle when the constraint is enabled, and dropped when the constraint is disabled or dropped. No action is required by the user in either case to manage the index. However, these associated indexes affect how you manage UNIQUE key and PRIMARY KEY constraints.

When disabling or dropping UNIQUE key and PRIMARY KEY integrity constraints, consider the following issues:

- The constraint's associated index will be dropped when the constraint is dropped or disabled.
- While enabled foreign keys reference a primary or unique key, you cannot disable or drop the primary or unique key constraint.

If the constraint is subsequently enabled or redefined, Oracle creates another index for the constraint.

Because unique and primary keys have associated indexes, you should factor in the cost of dropping and creating indexes when considering whether to disable or drop a UNIQUE or PRIMARY KEY constraint. If the associated index for a UNIQUE key or PRIMARY KEY constraint is extremely large, you may save time by leaving the constraint enabled rather than dropping and re–creating the large index.

Enabling and Disabling Integrity Constraints Upon Definition

When an integrity constraint is defined in a CREATE TABLE or ALTER TABLE statement, it can be enabled by including the ENABLE clause in the constraint's definition, or disabled by including the DISABLE clause in the constraint's definition. If neither the ENABLE nor DISABLE clause is included in a constraint's definition, Oracle automatically enables the constraint.

Enabling Constraints Upon Definition

The following CREATE TABLE and ALTER TABLE statements both define and enable integrity constraints:

```
CREATE TABLE emp (
   empno NUMBER(5) PRIMARY KEY,   . . . ;
ALTER TABLE emp
   ADD PRIMARY KEY (empno);
```

An ALTER TABLE statement that defines and attempts to enable an integrity constraint may fail because rows of the table may violate the integrity constraint. In this case, the statement is rolled back and the constraint definition is not stored and not enabled.

To enable a UNIQUE key or PRIMARY KEY, which creates an associated index, the owner of the table also needs a quota for the tablespace intended to contain the index, or the UNLIMITED TABLESPACE system privilege.

Disabling Constraints Upon Definition

The following CREATE TABLE and ALTER TABLE statements both define and disable integrity constraints:

```
CREATE TABLE emp (
   empno NUMBER(5) PRIMARY KEY DISABLE,   . . . ;

ALTER TABLE emp
   ADD PRIMARY KEY (empno) DISABLE;
```

An ALTER TABLE statement that defines and disables an integrity constraints never fails because of rows of the table that violate the integrity constraint. The definition of the constraint is allowed because its rule is not enforced.

See Also: For more information about constraint exceptions, see "Reporting Constraint Exceptions" on page 16 – 16.

Enabling and Disabling Existing Integrity Constraints

You can use the ALTER TABLE command with the ENABLE clause to enable a disabled constraint., or, with the DISABLE clause, to disable an enabled constraint.

Enabling Disabled Constraints

The following statements enable disabled integrity constraints:

```
ALTER TABLE dept
   ENABLE CONSTRAINT dname_ukey;
ALTER TABLE dept
   ENABLE PRIMARY KEY,
   ENABLE UNIQUE (dname, loc);
```

An ALTER TABLE statement that attempts to enable an integrity constraint may fail because rows of the table may violate the integrity constraint. In this case, the statement is rolled back and the constraint is not enabled.

To enable a UNIQUE key or PRIMARY KEY (which creates an associated index), the owner of the table also needs a quota for the tablespace intended to contain the index, or the UNLIMITED TABLESPACE system privilege.

Disabling Enabled Constraints

The following statements disable integrity constraints:

```
ALTER TABLE dept
   DISABLE CONSTRAINT dname_ukey;
ALTER TABLE dept
   DISABLE PRIMARY KEY,
   DISABLE UNIQUE (dname, loc);
```

To disable or drop a UNIQUE key or PRIMARY KEY constraint and all dependent FOREIGN KEY constraints in a single step, use the CASCADE option of the DISABLE or DROP clauses. For example, the following statement disables a PRIMARY KEY constraint and any FOREIGN KEY constraints that depend on it:

```
ALTER TABLE dept
   DISABLE PRIMARY KEY CASCADE;
```

See Also: For more information about constraint exceptions, see “Reporting Constraint Exceptions” on page 16 – 16.

Dropping Integrity Constraints

You can drop an integrity constraint if the rule that it enforces is no longer true, or if the constraint is no longer needed. You can drop the constraint using the ALTER TABLE command with the DROP clause. The following two statements drop integrity constraints:

```
ALTER TABLE dept
   DROP UNIQUE (dname, loc);
ALTER TABLE emp
   DROP PRIMARY KEY,
   DROP CONSTRAINT dept_fkey;
```

Dropping UNIQUE key and PRIMARY KEY constraints drops the associated indexes. Also, if FOREIGN KEYs reference a UNIQUE or PRIMARY KEY, you must include the CASCADE CONSTRAINTS clause in the DROP statement, or you cannot drop the constraint.

Reporting Constraint Exceptions

If no exceptions are present when a CREATE TABLE. . . ENABLE. . . or ALTER TABLE. . . ENABLE. . . statement is issued, the integrity constraint is enabled and all subsequent DML statements are subject to the enabled integrity constraints.

If exceptions exist when a constraint is enabled, an error is returned and the integrity constraint remains disabled. When a statement is not successfully executed because integrity constraint exceptions exist, the statement is rolled back. If exceptions exist, you cannot enable the constraint until all exceptions to the constraint are either updated or deleted.

To determine which rows violate the integrity constraint, issue the CREATE TABLE or ALTER TABLE statement with the EXCEPTIONS option in the ENABLE clause. The EXCEPTIONS option places the ROWID, table owner, table name, and constraint name of all exception rows into a specified table. For example, the following statement attempts to enable the PRIMARY KEY of the DEPT table, and if exceptions exist, information is inserted into a table named EXCEPTIONS:

```
ALTER TABLE dept ENABLE PRIMARY KEY EXCEPTIONS INTO exceptions;
```

> **Note:** You must create an appropriate exceptions report table to accept information from the EXCEPTIONS option of the ENABLE clause before enabling the constraint. You can create an exception table by submitting the script UTLEXCPT.SQL, which creates a table named EXCEPTIONS. You can create additional exceptions tables with different names by modifying and re–submitting the script.

If duplicate primary key values exist in the DEPT table and the name of the PRIMARY KEY constraint on DEPT is SYS_C00301, the following rows might be placed in the table EXCEPTIONS by the previous statement:

```
SELECT * FROM exceptions;

ROWID               OWNER      TABLE_NAME      CONSTRAINT
------------------  ---------  --------------  -----------
000003A5.000C.0001  SCOTT      DEPT            SYS_C00301
000003A5.000D.0001  SCOTT      DEPT            SYS_C00301
```

A more informative query would be to join the rows in an exception report table and the master table to list the actual rows that violate a specific constraint, as shown in the following example:

```
SELECT deptno, dname, loc FROM dept, exceptions
   WHERE exceptions.constraint = ’SYS_C00301’
   AND dept.rowid = exceptions.row_id;

DEPTNO     DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        10 RESEARCH       DALLAS
```

All rows that violate a constraint must be either updated or deleted from the table containing the constraint. When updating exceptions, you must change the value violating the constraint to a value consistent with the constraint or a null. After the row in the master table is updated or deleted, the corresponding rows for the exception in the exception report table should be deleted to avoid confusion with later exception reports. The statements that update the master table and the exception report table should be in the same transaction to ensure transaction consistency.

To correct the exceptions in the previous examples, you might issue the following transaction:

```
UPDATE dept SET deptno = 20 WHERE dname = ’RESEARCH’;
DELETE FROM exceptions WHERE constraint = ’SYS_C00301’;
COMMIT;
```

When managing exceptions, the goal is to eliminate all exceptions in your exception report table.

> **Note:** While you are correcting current exceptions for a table with the constraint disabled, other users may issue statements creating new exceptions.

See Also: The exact name and location of the UTLEXCPT.SQL script is operating system–specific. For more information, see your operating system–specific Oracle documentation.

Managing Object Dependencies

This section describes the various object dependencies, and includes the following topics:

- Manually Recompiling Views
- Manually Recompiling Procedures and Functions
- Manually Recompiling Packages

First, review Table 16 – 1, which shows how objects are affected by changes in other objects on which they depend.

Operation	Resulting Status of Object	Resulting Status of Dependent Objects
CREATE table, sequence, synonym	VALID if there are no errors	No change[1]
ALTER table (ADD column MODIFY column) RENAME table, sequence, synonym, view	VALID if there no errors	INVALID
DROP table, sequence, synonym, view, procedure, function, package	None; the object is dropped	INVALID
CREATE view, procedure[2]	VALID if there are no errors; INVALID if there are syntax or authorization errors	No change[1]
CREATE OR REPLACE view or procedure[2]	VALID if there are no error; INVALID if there are syntax or authorization errors	INVALID
REVOKE object privilege[3] ON object TO/FROM user	No change	All objects of user that depend on object are INVALID[3]
REVOKE object privilege[3] ON object TO/FROM PUBLIC	No change	All objects in the database that depend on object are INVALID[3]
REVOKE system privilege[4] TO/FROM user	No change	All objects of user are INVALID[4]
REVOKE system privilege[4] TO/FROM PUBLIC	No change	All objects in the database are INVALID[4]

Table 16 – 1 Operations that Affect Object Status

[1] *May cause dependent objects to be made INVALID, if object did not exist earlier.*

[2] *Stand–alone procedures and functions, packages, and triggers.*

[3] *Only DML object privileges, including SELECT, INSERT, UPDATE, DELETE, and EXECUTE; revalidation does not require recompiling.*

[4] *Only DML system privileges, including SELECT, INSERT, UPDATE, DELETE ANY TABLE, and EXECUTE ANY PROCEDURE; revalidation does not require recompiling.*

Oracle automatically recompiles an invalid view or PL/SQL program unit the next time it is used. In addition, a user can force Oracle to recompile a view or program unit using the appropriate SQL command with the COMPILE parameter. Forced compilations are most often used to test for errors when a dependent view or program unit is invalid, but is not currently being used. In these cases, automatic recompilation would not otherwise occur until the view or program unit was executed. To identify invalid dependent objects, query the views USER_/ALL_/DBA_OBJECTS.

Manually Recompiling Views

To recompile a view manually, the view must be contained in your schema or you must have the ALTER ANY TABLE system privilege. Use the ALTER VIEW command with the COMPILE parameter to recompile a view. The following statement recompiles the view EMP_DEPT contained in your schema:

```
ALTER VIEW emp_dept COMPILE;
```

Manually Recompiling Procedures and Functions

To recompile a procedure manually, the procedure must be contained in your schema, or you must have the ALTER ANY PROCEDURE system privilege. Use the ALTER PROCEDURE/FUNCTION command with the COMPILE parameter to recompile a stand–alone procedure or function. The following statement recompiles the stored procedure UPDATE_SALARY contained in your schema:

```
ALTER PROCEDURE update_salary COMPILE;
```

Manually Recompiling Packages

To recompile a package manually, the package must be contained in your schema, or you must have the ALTER ANY PROCEDURE system privilege. Use the ALTER PACKAGE command with the COMPILE parameter to recompile either a package body or both a package specification and body. The following statements recompile just the body, and the body and specification of the package ACCT_MGMT, respectively:

```
ALTER PACKAGE acct_mgmt COMPILE BODY;
ALTER PACKAGE acct_mgmt COMPILE PACKAGE;
```

Managing Object Name Resolution

Object names referenced in SQL statements can consist of several pieces, separated by periods. Oracle resolves an object name using the following algorithm:

1. Oracle attempts to qualify the first piece of the name referenced in the SQL statement. For example, in SCOTT.EMP, SCOTT is the first piece. If there is only one piece, the one piece is considered the first piece.

   1.1 In the current schema, Oracle searches for an object whose name matches the first piece of the object name. If it does not find such an object, it continues with Step 1.2.

   1.2 If no schema object is found in the current schema, Oracle searches for a public synonym that matches the first piece of the name. If it does not find one, it continues with Step 1.3.

   1.3 If no public synonym is found, Oracle searches for a schema whose name matches the first piece of the object name. If it finds one, it returns to Step 1.1, now using the second piece of the name as the object to find in the qualified schema. If the second piece does not correspond to a object in the previously qualified schema or there is not a second piece, Oracle returns an error.

   If no schema is found in Step 1.3, the object cannot be qualified and Oracle returns an error.

2. A schema object has been qualified. Any remaining pieces of the name must match a valid part of the found object. For example, if SCOTT.EMP.DEPTNO is the name, SCOTT is qualified as a schema, EMP is qualified as a table, and DEPTNO must correspond to a column (because EMP is a table). If EMP is qualified as a package, DEPTNO must correspond to a public constant, variable, procedure, or function of that package.

When global object names are used in a distributed database, either explicitly or indirectly within a synonym, the local Oracle resolves the reference locally. For example, it resolves a synonym to a remote table's global object name. The partially resolved statement is shipped to the remote database, and the remote Oracle completes the resolution of the object as described here.

Changing Storage Parameters for the Data Dictionary

This section describes aspects of changing data dictionary storage parameters, and includes the following topics:

- Structures in the Data Dictionary
- Errors that Require Changing Data Dictionary Storage

If your database is very large or contains an unusually large number of objects, columns in tables, constraint definitions, users, or other definitions, the tables that make up the data dictionary might at some point be unable to acquire additional extents. For example, a data dictionary table may need an additional extent, but there is not enough contiguous space in the SYSTEM tablespace. If this happens, you cannot create new objects, even though the tablespace intended to hold the objects seems to have sufficient space. To remedy this situation, you can change the storage parameters of the underlying data dictionary tables to allow them to be allocated more extents, in the same way that you can change the storage settings for user–created segments. For example, you can adjust the values of NEXT or PCTINCREASE for the data dictionary table.

⚠ **Warning:** Exercise caution when changing the storage settings for the data dictionary objects. If you choose inappropriate settings, you could damage the structure of the data dictionary and be forced to re–create your entire database. For example, if you set PCTINCREASE for the data dictionary table USER$ to 0 and NEXT to 2K, that table will quickly reach the maximum number of extents for a segment, and you will not be able to create any more users or roles without exporting, re–creating, and importing the entire database.

Structures in the Data Dictionary

The following tables and clusters contain the definitions of all the user–created objects in the database:

SEG$	segments defined in the database (including temporary segments)
OBJ$	user–defined objects in the database (including clustered tables); indexed by I_OBJ1 and I_OBJ2
UNDO$	rollback segments defined in the database; indexed by I_UNDO1
FET$	available free extents not allocated to any segment
UET$	extents allocated to segments
TS$	tablespaces defined in the database

FILE$	files that make up the database; indexed by I_FILE1
FILEXT$	datafiles with the AUTOEXTEND option set on
TAB$	tables defined in the database (includes clustered tables); indexed by I_TAB1
CLU$	clusters defined in the database
IND$	indexes defined in the database; indexed by I_IND1
ICOL$	columns that have indexes defined on them (includes individual entries for each column in a composite index); indexed by I_ICOL1
COL$	columns defined in tables in the database; indexed by I_COL1 and I_COL2
CON$	constraints defined in the database (includes information on constraint owner); indexed by I_CON1 and I_CON2
CDEF$	definitions of constraints in CON$; indexed by I_CDEF1, I_CDEF2, and I_CDEF3
CCOL$	columns that have constraints defined on them (includes individual entries for each column in a composite key); indexed by I_CCOL1
USER$	users and roles defined in the database; indexed by I_USER1
TSQ$	tablespace quotas for users (contains one entry for each tablespace quota defined for each user)
C_OBJ#	cluster containing TAB$, CLU$, ICOL$, IND$, and COL$: indexed by I_OBJ#
C_TS#	cluster containing FET$, TS$, and FILE$; indexed by I_TS#
C_FILE#_BLOCK#	cluster containing SEG$ and UET$; indexed by I_FILE#_BLOCK#
C_USER#	cluster containing USER and TSQ$$; indexed by I_USER#
C_COBJ#	cluster containing CDEF$ and CCOL$; indexed by I_COBJ#

Of all of the data dictionary segments, the following are the most likely to require changes:

C_TS#	if the free space in your database is very fragmented
C_OBJ#	if you have many indexes or many columns in your tables
CON$, C_COBJ#	if you use integrity constraints heavily
C_USER#	if you have a lot of users defined in your database

For the clustered tables, you must change the storage settings for the cluster, not for the table.

Errors that Require Changing Data Dictionary Storage

Oracle returns an error if a user tries to create a new object that requires Oracle to allocate an additional extent to the data dictionary when it is unable to allocate an extent. The error message ORA–1547, “failed to allocate extent of size *num* in tablespace ’*name*’” indicates this kind of problem.

If you receive this error message and the segment you were trying to change (such as a table or rollback segment) has not reached the limits specified for it in its definition, check the storage settings for the object that contains its definition.

For example, if you received an ORA–1547 while trying to define a new PRIMARY KEY constraint on a table and there is sufficient space for the index that Oracle must create for the key, check if CON$ or C_COBJ# cannot be allocated another extent; to do this, query DBA_SEGMENTS and consider changing the storage parameters for CON$ or C_COBJ#.

See Also: For more information, see “Displaying Segments that Cannot Allocate Additional Extents” on page 16 – 27.

Displaying Information About Schema Objects

The data dictionary provides many views about the schema objects described in chapters 10–16. The following list summarizes the views associated with schema objects:

- ALL_OBJECTS, USER_OBJECTS, DBA_OBJECTS
- ALL_CATALOG, USER_CATALOG, DBA_CATALOG
- ALL_TABLES, USER_TABLES, DBA_TABLES
- ALL_TAB_COLUMNS, USER_TAB_COLUMNS, DBA_TAB_COLUMNS
- ALL_TAB_COMMENTS, USER_TAB_COMMENTS

- ALL_COL_COMMENTS, USER_COL_COMMENTS, DBA_COL_COMMENTS
- ALL_VIEWS, USER_VIEWS, DBA_VIEWS
- ALL_INDEXES, USER_INDEXES, DBA_INDEXES
- ALL_IND_COLUMNS, USER_IND_COLUMNS, DBA_IND_COLUMNS
- USER_CLUSTERS, DBA_CLUSTERS
- USER_CLU_COLUMNS, DBA_CLU_COLUMNS
- ALL_SEQUENCES, USER_SEQUENCES, DBA_SEQUENCES
- ALL_SYNONYMS, USER_SYNONYMS, DBA_SYNONYMS
- ALL_DEPENDENCIES, USER_DEPENDENCIES, DBA_DEPENDENCIES

The following data dictionary views contain information about the segments of a database:

- USER_SEGMENTS
- DBA_SEGMENTS

The following data dictionary views contain information about a database's extents:

- USER_EXTENTS
- DBA_EXTENTS
- USER_FREE_SPACE
- DBA_FREE_SPACE

Oracle Packages

Table 16 – 2 describes packages that are supplied with Oracle to either allow PL/SQL access to some SQL features, or to extend the functionality of the database.

Procedure	Description
`dbms_space.unused_space`	Returns information about unused space in an object (table, index, or cluster).
`dbms_space.free_blocks`	Returns information about free blocks in an object (table, index, or cluster).

Table 16 – 2 Supplied Packages: Additional Functionality

Procedure	Description
dbms_session.free_unused_ user_memory	Procedure for reclaiming unused memory after performing operations requiring large amounts of memory (where large>100K). This procedure should only be used in cases where memory is at a premium.
dbms_system.set_sql_trace_in _session	Enables sql_trace in the session identified by serial number and SID (these values are located in v$session).

Table 16 – 2 Supplied Packages: Additional Functionality

The following examples demonstrate ways to display miscellaneous schema objects.

Example 1 Displaying Schema Objects By Type

The following query lists all of the objects owned by the user issuing the query:

```
SELECT object_name, object_type FROM user_objects;

OBJECT_NAME                OBJECT_TYPE
-------------------------  -------------------
EMP_DEPT                   CLUSTER
EMP                        TABLE
DEPT                       TABLE
EMP_DEPT_INDEX             INDEX
PUBLIC_EMP                 SYNONYM
EMP_MGR                    VIEW
```

Example 2 Displaying Column Information

Column information, such as name, datatype, length, precision, scale, and default data values can be listed using one of the views ending with the _COLUMNS suffix. For example, the following query lists all of the default column values for the EMP and DEPT tables:

```
SELECT table_name, column_name, data_default
   FROM user_tab_columns
   WHERE table_name = ’DEPT’ OR table_name = ’EMP’;

TABLE_NAME  COLUMN_NAME     DATA_DEFAULT
----------  --------------- --------------------
DEPT        DEPTNO
DEPT        DNAME
DEPT        LOC             ’NEW YORK’
EMP         EMPNO
EMP         ENAME
EMP         JOB
EMP         MGR
EMP         HIREDATE        SYSDATE
EMP         SAL
EMP         COMM
EMP         DEPTNO
```

Notice that not all columns have user–specified defaults. These columns automatically have NULL as the default.

Example 3 Displaying Dependencies of Views and Synonyms

When you create a view or a synonym, the view or synonym is based on its underlying base object. The ALL / USER / DBA_DEPENDENCIES data dictionary views can be used to reveal the dependencies for a view and the ALL / USER / DBA_SYNONYMS data dictionary views can be used to list the base object of a synonym. For example, the following query lists the base objects for the synonyms created by the user JWARD:

```
SELECT table_owner, table_name, synonym_name
   FROM sys.dba_synonyms
   WHERE owner = ’JWARD’;

TABLE_OWNER             TABLE_NAME   SYNONYM_NAME
----------------------- ------------ -----------------
SCOTT                   DEPT         DEPT
SCOTT                   EMP          EMP
```

Example 4 Displaying General Segment Information

The following query returns the name of each rollback segment, the tablespace that contains each, and the size of each rollback segment:

```
SELECT segment_name, tablespace_name, bytes, blocks, extents
   FROM sys.dba_segments
   WHERE segment_type = ’ROLLBACK’;

SEGMENT_NAME TABLESPACE_NAME      BYTES     BLOCKS    EXTENTS
------------ --------------- ---------- ---------- ----------
RS1          SYSTEM               20480         10          2
RS2          TS1                  40960         20          3
SYSTEM       SYSTEM              184320         90          3
```

Example 5 Displaying General Extent Information

General information about the currently allocated extents in a database is stored in the DBA_EXTENTS data dictionary view. For example, the following query identifies the extents associated with rollback segments and the size of each of those extents:

```
SELECT segment_name, bytes, blocks
   FROM sys.dba_extents
   WHERE segment_type = ’ROLLBACK’;

SEGMENT_NAME         BYTES     BLOCKS
--------------- ---------- ----------
RS1                  10240          5
RS1                  10240          5
SYSTEM               51200         25
SYSTEM               51200         25
SYSTEM               51200         25
```

Notice that the RS1 rollback segment is comprised of two extents, both 10K, while the SYSTEM rollback segment is comprised of three equally sized extents of 50K.

Example 6 Displaying the Free Space (Extents) of a Database

Information about the free extents (extents not allocated to any segment) in a database is stored in the DBA_FREE_SPACE data dictionary view. For example, the following query reveals the amount of free space available via free extents in each tablespace:

```
SELECT tablespace_name, file_id, bytes, blocks
   FROM sys.dba_free_space;

TABLESPACE_NAME         FILE_ID      BYTES     BLOCKS
-------------------- ---------- ---------- ----------
SYSTEM                        1    8120320       3965
SYSTEM                        1      10240          5
TS1                           2   10432512       5094
```

Example 7 Displaying Segments that Cannot Allocate Additional Extents

You can also use DBA_FREE_SPACE, in combination with the views DBA_SEGMENTS, DBA_TABLES, DBA_CLUSTERS, DBA_INDEXES, and DBA_ROLLBACK_SEGS, to determine if any other segment is unable to allocate additional extents for data dictionary objects only.

A segment may not be allocated to an extent for any of the following reasons:

- The tablespace containing the segment does not have enough room for the next extent.
- The segment has the maximum number of extents, as recorded in the data dictionary (in SEG.MAX_EXTENTS).
- The segment has the maximum number of extents allowed by the data block size, which is operating system specific.

Note: While the STORAGE clause value for MAXEXTENTS can be UNLIMITED, data dictionary tables cannot have MAXEXTENTS greater than the allowed block maximum. Thus, data dictionary tables cannot be converted to unlimited format.

The following query returns the names, owners, and tablespaces of all segments that fit any of the above criteria:

```
SELECT seg.owner, seg.segment_name,
   seg.segment_type, seg.tablespace_name,
   DECODE(seg.segment_type,
      'TABLE', t.next_extent,
      'CLUSTER', c.next_extent,
      'INDEX', i.next_extent,
      'ROLLBACK', r.next_extent)
```

```
FROM sys.dba_segments seg,
   sys.dba_tables t,
   sys.dba_clusters c,
   sys.dba_indexes i,
   sys.dba_rollback_segs r
WHERE ((seg.segment_type = 'TABLE'
   AND seg.segment_name = t.table_name
   AND seg.owner = t.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
      WHERE free.tablespace_name = t.tablespace_name
      AND free.bytes >= t.next_extent))
OR (seg.segment_type = 'CLUSTER'
   AND seg.segment_name = c.cluster_name
   AND seg.owner = c.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
      WHERE free.tablespace_name = c.tablespace_name
      AND free.bytes >= c.next_extent))
OR (seg.segment_type = 'INDEX'
   AND seg.segment_name = i.index_name
   AND seg.owner = i.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
      WHERE free.tablespace_name = i.tablespace_name
      AND free.bytes >= i.next_extent))
OR     (seg.segment_type = 'ROLLBACK'
   AND seg.segment_name = r.segment_name
   AND seg.owner = r.owner
   AND NOT EXISTS (SELECT tablespace_name
      FROM dba_free_space free
      WHERE free.tablespace_name = r.tablespace_name
     AND free.bytes >= r.next_extent)))
OR seg.extents = seg.max_extents OR seg.extents = data_block_size;
```

> **Note:** When you use this query, replace *data_block_size* with the data block size for your system.

Once you have identified a segment that cannot allocate additional extents, you can solve the problem in either of two ways, depending on its cause:

- If the tablespace is full, add datafiles to the tablespace.
- If the segment has too many extents, and you cannot increase MAXEXTENTS for the segment, perform the following steps: first, export the data in the segment; second, drop and recreate the segment, giving it a larger INITIAL setting so that it does not need to allocate so many extents; and third, import the data back into the segment.

CHAPTER

17

Managing Rollback Segments

This chapter describes how to manage rollback segments, and includes the following topics:

- Guidelines for Managing Rollback Segments
- Creating Rollback Segments
- Specifying Storage Parameters for Rollback Segments
- Taking Rollback Segments Online and Offline
- Explicitly Assigning a Transaction to a Rollback Segment
- Dropping Rollback Segments
- Monitoring Rollback Segment Information

See Also: If you are using Trusted Oracle7 in DBMS MAC mode, see the *Trusted Oracle7 Server Administrator's Guide* for additional information.

If you are using Oracle with the Parallel Server option, see the *Oracle7 Parallel Server Concepts & Administration* guide.

This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide.*

Guidelines for Managing Rollback Segments

This section describes guidelines to consider before creating or managing the rollback segments of your databases, and includes the following topics:

- Use Multiple Rollback Segments
- Choose Between Public and Private Rollback Segments
- Specify Rollback Segments to Acquire Automatically
- Set Rollback Segment Sizes Appropriately
- Create Rollback Segments with Many Equally–Sized Extents
- Set an Optimal Number of Extents for Each Rollback Segment
- Set the Storage Location for Rollback Segments

Every database contains one or more *rollback segments*, which are portions of the database that record the actions of transactions in the event that a transaction is rolled back. You use rollback segments to provide read consistency, rollback transactions, and recover the database.

See Also: For more information about rollback segments, see the *Oracle7 Server Concepts* manual.

Use Multiple Rollback Segments

Using multiple rollback segments distributes rollback segment contention across many segments and improves system performance. Multiple rollback segments are required in the following situations:

- When a database is created, a single rollback segment named SYSTEM is created in the SYSTEM tablespace. If a database is to have other tablespaces, it *must* have two or more rollback segments in the SYSTEM tablespace. You cannot create any objects in non–SYSTEM tablespaces (not even rollback segments) until you have created and brought online at least one additional rollback segment in the SYSTEM tablespace.
- When many transactions are concurrently proceeding, more rollback information is generated at the same time. You can indicate the number of concurrent transactions you expect for the instance with the parameter TRANSACTIONS, and the number of transactions you expect each rollback segment to have to handle with the parameter TRANSACTIONS_PER_–ROLLBACK_SEGMENT. Then, when an instance opens a database, it attempts to acquire at least TRANSACTIONS/ TRANSACTIONS_PER_ROLLBACK_SEGMENT rollback segments to handle the maximum amount of transactions.

Therefore, after setting the parameters, create TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_–SEGMENT rollback segments.

See Also: With the Oracle Parallel Server, in order to start, each instance requires access to its own rollback segment, in addition to the SYSTEM rollback segment. For additional details, see the *Oracle7 Parallel Server Concepts & Administration* guide.

Add a Rollback Segment to the SYSTEM Tablespace

An initial rollback segment called SYSTEM is created when a database is created. The SYSTEM rollback segment is created in the SYSTEM tablespace using the default storage parameters associated with that tablespace. You cannot drop this rollback segment.

An instance always acquires the SYSTEM rollback segment in addition to any other rollback segments it needs. However, if there are multiple rollback segments, Oracle tries to use the SYSTEM rollback segment only for special system transactions and distributes user transactions among other rollback segments; if there are too many transactions for the non–SYSTEM rollback segments, Oracle uses the SYSTEM segment. Therefore, after database creation, create at least one additional rollback segment in the SYSTEM tablespace.

Choose Between Public and Private Rollback Segments

A *private rollback segment* is acquired explicitly by an instance when the instance opens the database. *Public rollback segments* form a pool of rollback segments that any instance requiring a rollback segment can use.

If a database does not have the Parallel Server option, public and private rollback segments are identical. Therefore, you can create all public rollback segments. A database with the Parallel Server option can also have only public segments, as long as the number of segments is high enough that each instance opening the database can acquire at least one rollback segment in addition to its SYSTEM rollback segment. You may also use private rollback segments when using the Oracle Parallel Server.

See Also: For more information about the Parallel Server option and rollback segments, see the *Oracle7 Parallel Server Concepts & Administration* guide.

For more information about public and private rollback segments, see the *Oracle7 Server Concepts* guide.

Specify Rollback Segments to Acquire Automatically

By default, when an instance starts, it acquires TRANSACTIONS / TRANSACTIONS_PER_ROLLBACK_SEGMENT rollback segments. If you want to ensure that the instance acquires particular rollback segments that have particular sizes or particular tablespaces, specify the rollback segments by name in the ROLLBACK_SEGMENTS parameter in the instance's parameter file.

The instance acquires all the rollback segments listed in this parameter, even if more than TRANSACTIONS / TRANSACTIONS_PER_ROLL–BACK_SEGMENT segments are specified. The rollback segments can be either private or public.

Set Rollback Segment Sizes Appropriately

Total rollback segment size should be set based on the size of the most common transactions issued against a database. In general, short transactions experience better performance when the database has many smaller rollback segments, while long running transactions, like batch jobs, perform better with larger rollback segments. Generally, rollback segments can handle transactions of any size easily; however, in extreme cases when a transaction is either very short or very long, a user might want to use an appropriately sized rollback segment.

If a system is running only short transactions, rollback segments should be small so that they are always cached in main memory. If the rollback segments are small enough, they are more likely to be cached in the SGA according to the LRU algorithm, and database performance is improved because less disk I / O is necessary. The main disadvantage of small rollback segments is the increased likelihood of the error "snapshot too old" when running a long query involving records that are frequently updated by other transactions. This error occurs because the rollback entries needed for read consistency are overwritten as other update entries wrap around the rollback segment. Consider this issue when designing an application's transactions, and make them short atomic units of work so that you can avoid this problem.

In contrast, long running transactions work better with larger rollback segments, because the rollback entries for a long running transaction can fit in pre–allocated extents of a large rollback segment.

When a database system's applications concurrently issue a mix of very short and very long transactions, performance can be optimized if transactions are explicitly assigned to a rollback segment based on the transaction / rollback segment size. You can minimize dynamic extent allocation and truncation for rollback segments. This is not required for most systems and is intended for extremely large or small transactions.

To optimize performance when issuing a mix of extremely small and large transactions, make a number of rollback segments of appropriate size for each type of transaction (such as small, medium, and large). Most rollback segments should correspond to the typical transactions, with a fewer number of rollback segments for the atypical transactions. Then set OPTIMAL for each such rollback segment so that the rollback segment returns to its intended size if it has to grow.

You should tell users about the different sets of rollback segments that correspond to the different types of transactions. Often, it is *not* beneficial to assign a transaction explicitly to a specific rollback segment; however, you can assign an atypical transaction to an appropriate rollback segment created for such transactions. For example, you can assign a transaction that contains a large batch job to a large rollback segment.

When a mix of transactions is not prevalent, each rollback segment should be 10% of the size of the database's largest table because most SQL statements affect 10% or less of a table; therefore, a rollback segment of this size should be sufficient to store the actions performed by most SQL statements.

Generally speaking, you should set a high MAXEXTENTS for rollback segments; this allows a rollback segment to allocate subsequent extents as it needs them.

Create Rollback Segments with Many Equally–Sized Extents

Each rollback segment's total allocated space should be divided among many equally–sized extents. In general, optimal rollback I/O performance is observed if each rollback segment for an instance has 10 to 20 equally sized extents.

After determining the desired total initial size of a rollback segment and the number of initial extents for the segment, use the following formula to calculate the size of each extent of the rollback segment:

```
T / n = s
```

where:

T = total initial rollback segment size, in bytes

n = number of extents initially allocate

s = calculated size, in bytes, of each extent initially allocated

After s is calculated, create the rollback segment and specify the storage parameters INITIAL and NEXT as s, and MINEXTENTS to n. PCTINCREASE cannot be specified for rollback segments and therefore defaults to 0. Also, if the size s of an extent is not an exact multiple of the data block size, it is rounded up to the next multiple.

Set an Optimal Number of Extents for Each Rollback Segment

You should carefully assess the kind of transactions the system runs when setting the OPTIMAL parameter for each rollback segment. For a system that executes long–running transactions frequently, OPTIMAL should be large so that Oracle does not have to shrink and allocate extents frequently. Also, for a system that executes long queries on active data, OPTIMAL should be large to avoid "snapshot too old" errors. OPTIMAL should be smaller for a system that mainly executes short transactions and queries so that the rollback segments remain small enough to be cached in memory, thus improving system performance.

To obtain estimates and monitor the effectiveness of the OPTIMAL settings for rollback segments, use the MONITOR ROLLBACK feature of Server Manager / GUI. In this monitor, the following statistics are given for each rollback segment:

Sizes, High Water — the most space ever allocated for the rollback segment, in bytes

Sizes, Optimal — the OPTIMAL size of the rollback segment, in bytes

Occurrences, Wraps — the cumulative number of times a transaction continues writing from one extent in a rollback segment to another existing extent

Occurrences, Extends — the cumulative number of times a new extent is allocated for a rollback segment

Shrinks — the cumulative number of times Oracle has truncated extents from the rollback segment

Average Sizes, Shrunk — the average size of the space Oracle truncated from the rollback segment, in bytes

Average Sizes, Active — the average number of bytes in active extents in the rollback segment, measured over time

Assuming that an instance has equally sized rollback segments with comparably sized extents, the OPTIMAL parameter for a given rollback segment should be set slightly higher than *Average Sizes, Active.*

Table 17 – 1 provides additional information on how to interpret the statistics given in this monitor.

Shrinks	**Average Sizes, Shrunk**	**Analysis and Recommendation**
Low	Low	If *Average Sizes, active* is close to *Sizes*, *Optimal,* then the OPTIMAL setting is correct. Otherwise, OPTIMAL is too large (not many shrinks are being performed.)
Low	High	Excellent: a good setting for OPTIMAL.
High	Low	OPTIMAL is too small: too many shrinks are being performed.
High	High	Periodic long transactions are probably causing these statistics. Set the OPTIMAL parameter higher until *Shrinks* is low.

Table 17 – 1 Analyzing the Effectiveness of Current OPTIMAL Settings

Set the Storage Location for Rollback Segments

If possible, create one tablespace specifically to hold all rollback segments, in addition to the two required in the SYSTEM tablespace. This way, all rollback segment data is stored separately from other types of data. Creating this "rollback segment" tablespace can provide the following benefits:

- A tablespace holding rollback segments can always be kept online, thus maximizing the combined storage capacity of rollback segments at all times. Note that if some rollback segments are not available, the overall database operation can be affected.
- Because tablespaces with active rollback segments cannot be taken offline, designating a tablespace to hold all rollback segments of a database ensures that the data stored in other tablespaces can be taken offline without concern for the database's rollback segments.
- A tablespace's free extents are likely to be more fragmented if the tablespace contains rollback segments that frequently allocate and deallocate extents.

Creating Rollback Segments

To create rollback segments, you must have the CREATE ROLLBACK SEGMENT system privilege. To create additional rollback segments for a database, use either the Create Rollback Segment property sheet of Server Manager, or the SQL command CREATE ROLLBACK SEGMENT. The tablespace to contain the new rollback segment must be online.

The following statement creates a public rollback segment named USERS_RS in the USERS tablespace, using the default storage parameters of the USERS tablespace:

```
CREATE PUBLIC ROLLBACK SEGMENT users_rs TABLESPACE users;
```

Bringing New Rollback Segments Online

If you create a private rollback segment, you should add the name of this new rollback segment to the ROLLBACK_SEGMENTS parameter in the parameter file for the database. Doing so enables the private rollback segment to be captured by the instance at instance startup. For example, if two new private rollback segments are created and named RS1 and RS2, the ROLLBACK_SEGMENTS parameter of the parameter file should be similar to the following:

```
ROLLBACK SEGMENTS= (RS1, RS2)
```

See Also: Once a rollback segment is created, it is not available for use by transactions of any instance until it is brought online. See “Taking Rollback Segments Online and Offline” on page 17 – 10 for more information.

Specifying Storage Parameters for Rollback Segments

This section describes aspects of specifying rollback segment storage parameters, and includes the following topics:

- Setting Storage Parameters When Creating a Rollback Segment
- Changing Rollback Segment Storage Parameters
- Altering Rollback Segment Format
- Shrinking a Rollback Segment Manually

Setting Storage Parameters When Creating a Rollback Segment

Suppose you wanted to create a public rollback segment DATA1_RS with storage parameters and optimal size set as follows:

- The rollback segment is allocated an initial extent of 50K.
- The rollback segment is allocated the second extent of 50K.
- The optimal size of the rollback segment is 750K.
- The minimum number of extents and the number of extents initially allocated when the segment is created is 15.
- The maximum number of extents that the rollback segment can allocate, including the initial extent, is 100.

The following statement creates a rollback segment with these characteristics:

```
CREATE PUBLIC ROLLBACK SEGMENT data1_rs
   TABLESPACE users
   STORAGE (
      INITIAL 50K
      NEXT 50K
      OPTIMAL 750K
      MINEXTENTS 15
      MAXEXTENTS 100);
```

You can also use the Create Rollback Segment property sheet of Server Manager to set the rollback segment's storage parameters.

Changing Rollback Segment Storage Parameters

You can change a rollback segment's storage parameters after creating it. However, you cannot alter the size of any extent currently allocated to a rollback segment. You can only affect future extents.

Alter a rollback segment's storage parameters using either the Alter Rollback Segment property sheet of Server Manager, or the SQL command ALTER ROLLBACK SEGMENT.

The following statement alters the maximum number of extents that the DATA1_RS rollback segment can allocate.

```
ALTER PUBLIC ROLLBACK SEGMENT data1_rs
   STORAGE (MAXEXTENTS 120);
```

You can alter the settings for the SYSTEM rollback segment, including the OPTIMAL parameter, just as you can alter those of any rollback segment.

> **Note:** If you are altering a public rollback segment, you must include the keyword PUBLIC in the ALTER ROLLBACK SEGMENT command.

See Also: For guidance on setting sizes and storage parameters (including OPTIMAL) for rollback segments, see "Guidelines for Managing Rollback segments" on page 17 – 2.

Altering Rollback Segment Format

To alter rollback segments, you must have the ALTER ROLLBACK SEGMENT system privilege.

You can define limited or unlimited format for rollback segments. When converting to limited or unlimited format, you *must* take the rollback segments offline. If you identify unlimited format for rollback segments, extents for that segment must have a minimum of 4 data blocks. Thus, a limited format rollback segment cannot be converted to unlimited format if it has less than 4 data blocks in any extent. If you want to convert from limited to unlimited format and have less than 4 data blocks in an extent, your only choice is to drop and re–create the rollback segment.

Shrinking a Rollback Segment Manually

To shrink a rollback segment using you must have the ALTER ROLLBACK SEGMENT system privilege.

You can manually decrease the size of a rollback segment using the SQL command ALTER ROLLBACK SEGMENT. The rollback segment you are trying shrink must be online.

The following statement shrinks rollback segment RBS1 to 100K:

```
ALTER ROLLBACK SEGMENT rbs1 SHRINK TO 100K;
```

See Also: For a complete description of the ALTER ROLLBACK SEGMENT command, see the *Oracle7 Server SQL Reference.*

Taking Rollback Segments Online and Offline

This section describes aspects of taking rollback segments online and offline, and includes the following topics:

- Bringing Rollback Segments Online
- Taking Rollback Segments Offline

A rollback segment is either *online* and available to transactions, or *offline* and unavailable to transactions. Generally, rollback segments are online and available for use by transactions.

To take a rollback segment online or offline, you must have the ALTER ROLLBACK SEGMENT system privilege.

You may wish to take online rollback segments offline in the following situations:

- When you want to take a tablespace offline, and the tablespace contains rollback segments. You cannot take a tablespace offline if it contains rollback segments that transactions are currently using. To prevent associated rollback segments from being used, you can take them offline before taking the tablespace offline.
- You want to drop a rollback segment, but cannot because transactions are currently using it. To prevent the rollback segment from being used, you can take it offline before dropping it.

Note: You cannot take the SYSTEM rollback segment offline.

You might later want to bring an offline rollback segment back online so that transactions can use it. When a rollback segment is created, it is initially offline, and you must explicitly bring a newly created rollback segment online before it can be used by an instance's transactions. You can bring an offline rollback segment online via any instance accessing the database that contains the rollback segment.

Bringing Rollback Segments Online

You can bring online only a rollback segment whose current status (as shown in the DBA_ROLLBACK_SEGS data dictionary view) is OFFLINE or PARTLY AVAILABLE. To bring an offline rollback segment online, use either the Place Online menu item of Server Manager or the SQL command ALTER ROLLBACK SEGMENT with the ONLINE option.

Bringing a PARTLY AVAILABLE Rollback Segment Online

A rollback segment in the PARTLY AVAILABLE state contains data for an in–doubt or recovered distributed transaction, and yet to be recovered transactions. You can view its status in the data dictionary view DBA_ROLLBACK_SEGS as PARTLY AVAILABLE. The rollback segment usually remains in this state until the transaction is resolved either automatically by RECO, or manually by a DBA. However, you might find that all rollback segments are PARTLY AVAILABLE. In this case, you can bring a PARTLY AVAILABLE segment online, as described above.

Some resources used by the rollback segment for the in–doubt transaction remain inaccessible until the transaction is resolved. As a result, the rollback segment may have to grow if other transactions assigned to it need additional space.

As an alternative to bringing a PARTLY AVAILABLE segment online, you might find it easier to create a new rollback segment temporarily, until the in–doubt transaction is resolved.

Bringing a Rollback Segment Online Automatically

If you would like a rollback segment to be automatically brought online whenever you start up the database, add the segment's name to the ROLLBACK_SEGMENTS parameter in the database's parameter file.

Bringing Rollback Segments Online: Example

The following statement brings the rollback segment USER_RS_2 online:

```
ALTER ROLLBACK SEGMENT user_rs_2 ONLINE;
```

After you bring a rollback segment online, its status in the data dictionary view DBA_ROLLBACK_SEGS is ONLINE.

See Also: For information about the ROLLBACK_SEGMENTS and DBA_ROLLBACK_SEGS parameters, see the *Oracle7 Server Reference.*

To see a query for checking rollback segment state, see "Displaying Rollback Segment Information" on page 17 – 15.

Taking Rollback Segments Offline

To take an online rollback segment offline, use either the Take Offline menu item of Server Manager, or the ALTER ROLLBACK SEGMENT command with the OFFLINE option. The rollback segment's status in the DBA_ROLLBACK_SEGS data dictionary view must be "ONLINE", and the rollback segment must be acquired by the current instance.

The following example takes the rollback segment USER_RS_2 offline:

```
ALTER ROLLBACK SEGMENT user_rs_2 OFFLINE;
```

If you try to take a rollback segment that does not contain active rollback entries offline, Oracle immediately takes the segment offline and changes its status to "OFFLINE".

In contrast, if you try to take a rollback segment that contains rollback data for active transactions (local, remote, or distributed) offline, Oracle makes the rollback segment unavailable to future transactions and takes it offline after all the active transactions using the rollback segment complete. Until the transactions complete, the rollback segment cannot be brought online by any instance other than the one that was trying to take it offline. During this period, the rollback segment's status in the view DBA_ROLLBACK_SEGS remains ONLINE; however, the rollback segment's status in the view V$ROLLSTAT is PENDING OFFLINE.

The instance that tried to take a rollback segment offline and caused it to change to PENDING OFFLINE can bring it back online at any time; if the rollback segment is brought back online, it will function normally.

Taking Public and Private Rollback Segments Offline

After you take a public or private rollback segment offline, it remains offline until you explicitly bring it back online *or* you restart the instance.

See Also: For information on viewing rollback segment status, see "Displaying Rollback Segment Information" on page 17 – 15.

For information about the views DBA_ROLLBACK_SEGS and V$ROLLSTAT, see the *Oracle7 Server Reference*.

Explicitly Assigning a Transaction to a Rollback Segment

A transaction can be explicitly assigned to a specific rollback segment using the SET TRANSACTION command with the USE ROLLBACK SEGMENT parameter. Transactions are explicitly assigned to rollback segments for the following reasons:

- The anticipated amount of rollback information generated by a transaction can fit in the current extents of the assigned rollback segment.
- Additional extents do not have to be dynamically allocated (and subsequently truncated) for rollback segments, which reduces overall system performance.

No special privileges are required to assign a transaction to a specific rollback segment explicitly.

To assign a transaction to a rollback segment explicitly, the rollback segment must be online for the current instance, and the SET TRANSACTION USE ROLLBACK SEGMENT statement must be the first statement of the transaction. If a specified rollback segment is not online or a SET TRANSACTION USE ROLLBACK SEGMENT statement is not the first statement in a transaction, an error is returned.

For example, if you are about to begin a transaction that contains a significant amount of work (more than most transactions), you can assign the transaction to a large rollback segment, as follows:

```
SET TRANSACTION USE ROLLBACK SEGMENT large_rs1;
```

After the transaction is committed, Oracle will automatically assign the next transaction to any available rollback segment unless the new transaction is explicitly assigned to a specific rollback segment by the user.

Dropping Rollback Segments

You can drop rollback segments when the extents of a segment become too fragmented on disk, or the segment needs to be relocated in a different tablespace.

Before dropping a rollback segment, make sure that status of the rollback segment is OFFLINE. If the rollback segment that you want to drop is currently ONLINE, PARTLY AVAILABLE, NEEDS RECOVERY, or INVALID, you cannot drop it. If the status is INVALID, the segment has already been dropped. Before you can drop it, you must take it offline.

To drop a rollback segment, you must have the DROP ROLLBACK SEGMENT system privilege.

If a rollback segment is offline, you can drop it using either the Drop menu item of Server Manager, or the SQL command DROP ROLLBACK SEGMENT.

The following statement drops the DATA1_RS rollback segment:

```
DROP PUBLIC ROLLBACK SEGMENT data1_rs;
```

If you use the DROP ROLLBACK SEGMENT command, indicate the correct type of rollback segment to drop, public or private, by including or omitting the PUBLIC keyword.

> **Note:** If a rollback segment specified in ROLLBACK_SEGMENTS is dropped, make sure to edit the parameter files of the database to remove the name of the dropped rollback segment from the list in the ROLLBACK_SEGMENTS parameter. If this step is not performed before the next instance startup, startup fails because it cannot acquire the dropped rollback segment.

After a rollback segment is dropped, its status changes to INVALID. The next time a rollback segment is created, it takes the row vacated by a dropped rollback segment, if one is available, and the dropped rollback segment's row no longer appears in the DBA_ROLLBACK_SEGS view.

See Also: For more information about the view DBA_ROLLBACK_SEGS, see the *Oracle7 Server Reference.*

Monitoring Rollback Segment Information

Use the MONITOR ROLLBACK feature of Server Manager / GUI to monitor a rollback segment's size, number of extents, optimal number of extents, activity concerning dynamic deallocation of extents, and current usage by active transaction.

See Also: For a detailed description of how to use the MONITOR for the corresponding operation, see "Set an Optimal Number of Extents for Each Rollback Segment" on page 17 – 5.

Displaying Rollback Segment Information

The DBA_ROLLBACK_SEGS data dictionary view stores information about the rollback segments of a database. For example, the following query lists the name, associated tablespace, and status of each rollback segment in a database:

```
SELECT segment_name, tablespace_name, status
   FROM sys.dba_rollback_segs;

SEGMENT_NAME  TABLESPACE_NAME  STATUS
------------- ---------------- ------
SYSTEM        SYSTEM           ONLINE
PUBLIC_RS     SYSTEM           ONLINE
USERS_RS      USERS            ONLINE
```

In addition, the following data dictionary views contain information about the segments of a database, including rollback segments:

- USER_SEGMENTS
- DBA_SEGMENTS

Displaying All Rollback Segments

The following query returns the name of each rollback segment, the tablespace that contains it, and its size:

```
SELECT segment_name, tablespace_name, bytes, blocks, extents
   FROM sys.dba_segments
   WHERE segment_type = 'ROLLBACK';

SEGMENT_NAME TABLESPACE_NAME     BYTES      BLOCKS     EXTENTS
------------ --------------- --------- ---------- ----------
RS1          SYSTEM              20480         10          2
RS2          TS1                 40960         20          3
SYSTEM       SYSTEM             184320         90          3
```

Displaying Whether a Rollback Segment Has Gone Offline

When you take a rollback segment offline, it does not actually go offline until all active transactions in it have completed. Between the time when you attempt to take it offline and when it actually is offline, its status in DBA_ROLLBACK_SEGS remains ONLINE, but it is not used for new transactions. To determine whether any rollback segments for an instance are in this state, use the following query:

```
SELECT name, xacts ’ACTIVE TRANSACTIONS’
   FROM v$rollname, v$rollstat
WHERE status = ’PENDING OFFLINE’
   AND v$rollname.usn = v$rollstat.usn;

NAME       ACTIVE TRANSACTIONS
---------- -------------------
RS2                          3
```

If your instance is part of a Parallel Server configuration, this query displays information for rollback segments of the current instance only, not those of other instances.

Displaying Deferred Rollback Segments

The following query shows which rollback segments are private and which are public. Note that it only displays information about the rollback segments that are currently online for the current instance:

```
SELECT segment_name, tablespace_name, owner
   FROM sys.dba_rollback_segs;

SEGMENT_NAME  TABLESPACE_NAME  OWNER
------------- ---------------- ------
SYSTEM        SYSTEM           SYS
PUBLIC_RS     SYSTEM           PUBLIC
USERS_RS      USERS            SYS
```

Displaying All Deferred Rollback Segments

The following query shows all deferred rollback segments (rollback segments that were created to hold rollback entries for tablespaces taken offline until the tablespaces are brought back online):

```
SELECT segment_name, segment_type, tablespace_name
   FROM sys.dba_segments
WHERE segment_type = ’DEFERRED ROLLBACK’;

SEGMENT_NAME   SEGMENT_TYPE           TABLESPACE_NAME
------------   ------------           ---------------
USERS_RS       DEFERRED ROLLBACK      USERS
```

PART

IV

Database Security

CHAPTER

18

Establishing Security Policies

This chapter provides guidelines for developing security policies for database operation, and includes the following topics:

- System Security Policy
- Data Security Policy
- User Security Policy
- Auditing Policy

See Also: For information about additional security issues when you are using Trusted Oracle7, see the *Trusted Oracle7 Server Administrator's Guide.*

System Security Policy

This section describes aspects of system security policy, and includes the following topics:

- Database User Management
- User Authentication
- Operating System Security

Each database has one or more administrators who are responsible for maintaining all aspects of the security policy: the security administrators. If the database system is small, the database administrator may have the responsibilities of the security administrator. However, if the database system is large, a special person or group of people may have responsibilities limited to those of a security administrator.

After deciding who will manage the security of the system, a security policy must be developed for every database. A database's security policy should include several sub–policies, as explained in the following sections.

Database User Management

Database users are the access paths to the information in an Oracle database. Therefore, tight security should be maintained for the management of database users. Depending on the size of a database system and the amount of work required to manage database users, the security administrator may be the only user with the privileges required to create, alter, or drop database users. On the other hand, there may be a number of administrators with privileges to manage database users. Regardless, only trusted individuals should have the powerful privileges to administer database users.

User Authentication

Database users can be *authenticated* (verified as the correct person) by Oracle using the host operating system, network services, or the database. Generally, user authentication via the host operating system is preferred for the following reasons:

- Users can connect to Oracle faster and more conveniently without specifying a username or password.
- Centralized control over user authorization in the operating system: Oracle need not store or manage user passwords and usernames if the operating system and database correspond.
- User entries in the database and operating system audit trails correspond.

User authentication by the database is normally used when the host operating system cannot support user authentication.

See Also: For more information about network authentication, see *Oracle7 Server Distributed Systems, Volume I.*

For more information about user authentication, see "Creating Users" on page 19 – 9.

Operating System Security

If applicable, the following security issues must also be considered for the operating system environment executing Oracle and any database applications:

- Database administrators must have the operating system privileges to create and delete files.
- Typical database users should not have the operating system privileges to create or delete files related to the database.
- If the operating system identifies database roles for users, the security administrators must have the operating system privileges to modify the security domain of operating system accounts.

See Also: For more information about operating system security issues for Oracle databases, see your operating system–specific Oracle documentation.

Data Security Policy

Data security includes the mechanisms that control the access and use of the database at the object level. Your data security policy determines which users have access to a specific schema object, and the specific types of actions allowed for each user on the object. For example, user SCOTT can issue SELECT and INSERT statements but not DELETE statements using the EMP table. Your data security policy should also define the actions, if any, that are audited for each schema object.

Your data security policy will be determined primarily by the level of security you wish to establish for the data in your database. For example, it may be acceptable to have little data security in a database when you wish to allow any user to create any schema object, or grant access privileges for their objects to any other user of the system. Alternatively, it might be necessary for data security to be very controlled when you wish to make a database or security administrator the only person with the privileges to create objects and grant access privileges for objects to roles and users.

Overall data security should be based on the sensitivity of data. If information is not sensitive, then the data security policy can be more lax. However, if data is sensitive, a discreet security policy should be developed to maintain tight control over access to objects.

User Security Policy

This section describes aspects of user security policy, and includes the following topics:

- General User Security
- End–User Security
- Administrator Security
- Application Developer Security
- Application Administrator Security

General User Security

For all types of database users, consider the following general user security issues:

- Password Security
- Privilege Management

Password Security

If user authentication is managed by the database, security administrator's should develop a password security policy to maintain database access security. For example, database users should be required to change their passwords at regular intervals, and of course, when their passwords are revealed to others. By forcing a user to modify passwords in such situations, unauthorized database access can be reduced.

Secure Connections with Encrypted Passwords

To better protect the confidentiality of your password, Oracle7 can be configured to use encrypted passwords for client/server and server/server connections.

You can require that the password used to verify a connection always be encrypted by setting the following values:

- Set the ORA_ENCRYPT_LOGIN environment variable to TRUE on the client machine.
- Set the DBLINK_ENCRYPT_LOGIN server initialization parameter to TRUE.

If enabled at both the client and server, passwords will not be sent across the network "in the clear", but will be encrypted using a modified DES (Data Encryption Standard) algorithm.

The DBLINK_ENCRYPT_LOGIN parameter is used for connections between two Oracle servers (for example, when performing distributed queries). If you are connecting from a client, Oracle checks the ORA_ENCRYPT_LOGIN environment variable.

Whenever you attempt to connect to a server using a password, Oracle encrypts the password before sending it to the server. If the connection fails and auditing is enabled, the failure is noted in the audit log. Oracle then checks the appropriate DBLINK_ENCRYPT_LOGIN or ORA_ENCRYPT_LOGIN value. If it set to FALSE, Oracle attempts the connection again using an unencrypted version of the password. If the connection is successful, the connection replaces the previous failure in the audit log, and the connection proceeds. To prevent malicious users from forcing Oracle to re–attempt a connection with an unencrypted version of the password, you must set the appropriate values to TRUE.

Privilege Management

Security administrators should consider issues related to privilege management for all types of users. For example, in a database with many usernames, it may be beneficial to use roles (which are named groups of related privileges that you grant to users or other roles) to manage the privileges available to users. Alternatively, in a database with a handful of usernames, it may be easier to grant privileges explicitly to users and avoid the use of roles.

Security administrators managing a database with many users, applications, or objects should take advantage of the benefits offered by roles. Roles greatly simplify the task of privilege management in complicated environments.

End–User Security

Security administrators must also define a policy for end–user security. If a database is large with many users, the security administrator can decide what groups of users can be categorized, create user roles for these user groups, grant the necessary privileges or application roles to each user role, and assign the user roles to the users. To account for exceptions, the security administrator must also decide what privileges must be explicitly granted to individual users.

Using Roles for End–User Privilege Management

Roles are the easiest way to grant and manage the common privileges needed by different groups of database users.

Consider a situation where every user in the accounting department of a company needs the privileges to run the ACCTS_RECEIVABLE and ACCTS_PAYABLE database applications. Roles are associated with both applications, and contain the object privileges necessary to execute those applications.

The following actions, performed by the database or security administrator, address this simple security situation:

1. Create a role named ACCOUNTANT.
2. Grant the roles for the ACCTS_RECEIVABLE and ACCTS_PAYABLE database applications to the ACCOUNTANT role.
3. Grant each user of the accounting department the ACCOUNTANT role.

This security model is illustrated in Figure 18 – 1.

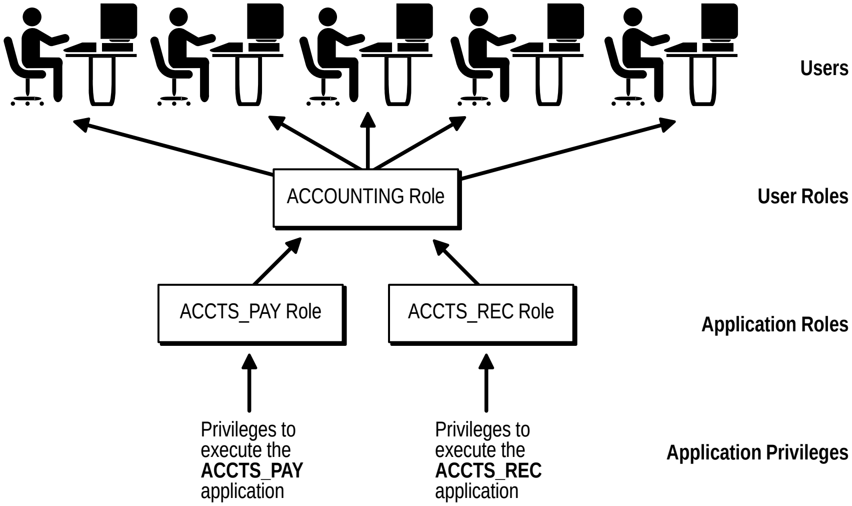

Users
ACCOUNTING Role
User Roles
ACCTS_PAY Role
ACCTS_REC Role
Application Roles
Privileges to execute the **ACCTS_PAY** application
Privileges to execute the **ACCTS_REC** application
Application Privileges

Figure 18 – 1 User Roles

This plan addresses the following potential situations:

- If accountants subsequently need a role for a new database application, that application's role can be granted to the ACCOUNTANT role, and all users in the accounting department will automatically receive the privileges associated with the new database application. The application's role does not need to be granted to individual users requiring use of the application.
- Similarly, if the accounting department no longer requires the need for a specific application, the application's role can be dropped from the ACCOUNTANT role.
- If the privileges required by the ACCTS_RECEIVABLE or ACCTS_PAYABLE applications change, the new privileges can be

granted to, or revoked from, the application's role. The security domain of the ACCOUNTANT role, and all users granted the ACCOUNTANT role automatically reflect the privilege modification.

When possible, utilize roles in all possible situations to make end–user privilege management efficient and simple.

Administrator Security

Security administrators should have a policy addressing administrator security. For example, when the database is large and there are several types of database administrators, the security administrator may decide to group related administrative privileges into several administrative roles. The administrative roles can then be granted to appropriate administrator users. Alternatively, when the database is small and has only a few administrators, it may be more convenient to create one administrative role and grant it to all administrators.

Protection for Connections as SYS and SYSTEM

After database creation, *immediately* change the passwords for the administrative SYS and SYSTEM usernames to prevent unauthorized access to the database. Connecting as SYS and SYSTEM give a user the powerful privileges to modify a database in many ways. Therefore, privileges for these usernames are extremely sensitive, and should only be available to select database administrators.

See Also: The passwords for these accounts can be modified using the procedures described in "Altering Users" on page 19 – 12.

Protection for Administrator Connections

Only database administrators should have the capability to connect to a database with administrator privileges. Connecting as SYSDBA or SYSOPER gives a user unrestricted privileges to do anything to a database (such as startup, shutdown, and recover) or the objects within a database (such as create, drop, and delete from).

Using Roles for Administrator Privilege Management

Roles are the easiest way to restrict the powerful system privileges and roles required by personnel administrating of the database.

Consider a scenario where the database administrator responsibilities at a large installation are shared among several database administrators, each responsible for the following specific database management jobs:

- an administrator responsible for object creation and maintenance
- an administrator responsible for database tuning and performance
- a security administrator responsible for creating new users, granting roles and privileges to database users

- a database administrator responsible for routine database operation (for example, startup, shutdown, backup)
- an administrator responsible for emergency situations, such as database recovery
- new, inexperienced database administrators needing limited capabilities to experiment with database management

In this scenario, the security administrator should structure the security for administrative personnel as follows:

1. Six roles should be defined to contain the distinct privileges required to accomplish each type of job (for example, DBA_OBJECTS, DBA_TUNE, DBA_SECURITY, DBA_MAINTAIN, DBA_RECOV, DBA_NEW).
2. Each role is granted the appropriate privileges.
3. Each type of database administrator can be granted the corresponding role.

This plan diminishes the likelihood of future problems in the following ways:

- If a database administrator's job description changes to include more responsibilities, that database administrator can be granted other administrative roles corresponding to the new responsibilities.
- If a database administrator's job description changes to include fewer responsibilities, that database administrator can have the appropriate administrative roles revoked.
- The data dictionary always stores information about each role and each user, so information is available to disclose the task of each administrator.

Application Developer Security

Security administrators must define a special security policy for the application developers using a database. A security administrator may grant the privileges to create necessary objects to application developers. Alternatively, the privileges to create objects may only be granted to a database administrator, who receives requests for object creation from developers.

Application Developers and Their Privileges

Database application developers are unique database users who require special groups of privileges to accomplish their jobs. Unlike end–users, developers need system privileges, such as CREATE TABLE, CREATE PROCEDURE, and so on. However, only specific system privileges should be granted to developers to restrict their overall capabilities in the database.

The Application Developer's Environment: Test and Production Databases

In many cases, application development is restricted to test databases and not allowed on production databases. This restriction ensures that application developers do not compete with end–users for database resources, and that they cannot detrimentally affect a production database.

After an application has been thoroughly developed and tested, it is permitted access to the production database and made available to the appropriate end–users of the production database.

Free Versus Controlled Application Development

The database administrator can define the following options when determining which privileges should be granted to application developers:

Free Development: An application developer is allowed to create new schema objects, including tables, indexes, procedures, packages, and so on. This option allows the application developer to develop an application independent of other objects.

Controlled Development: An application developer is not allowed to create new schema objects. All required tables, indexes, procedures, and so on are created by a database administrator, as requested by an application developer. This option allows the database administrator to completely control a database's space usage and the access paths to information in the database.

Although some database systems use only one of these options, other systems could mix them. For example, application developers can be allowed to create new stored procedures and packages, but not allowed to create tables or indexes. A security administrator's decision regarding this issue should be based on the following:

- the control desired over a database's space usage
- the control desired over the access paths to schema objects
- the database used to develop applications—if a test database is being used for application development, a more liberal development policy would be in order

Roles and Privileges for Application Developers

Security administrators can create roles to manage the privileges required by the typical application developer. For example, a typical role named APPLICATION_DEVELOPER might include the CREATE TABLE, CREATE VIEW, and CREATE PROCEDURE system privileges. Consider the following when defining roles for application developers:

- CREATE system privileges are usually granted to application developers so that they can create their own objects. However, CREATE ANY system privileges, which allow a user to create an object in any user's domain, are not usually granted to developers. This restricts the creation of new objects only to the developer's user account.
- Object privileges are rarely granted to roles used by application developers. This is often impractical because granting object privileges via roles often restricts their usability in the creation of other objects (primarily views and stored procedures). It is more practical to allow application developers to create their own objects for development purposes.

Space Restrictions Imposed on Application Developers

While application developers are typically given the privileges to create objects as part of the development process, security administrators must maintain limits on what and how much database space can be used by each application developer. For example, as the security administrator, you should specifically set or restrict the following limits for each application developer:

- the tablespaces in which the developer can create tables or indexes
- the quota for each tablespace accessible to the developer

See Also: Both limitations can be set by altering a developer's security domain. For more information, see "Altering Users" on page 19 – 12.

Application Administrator Security

In large database systems with many database applications (for example, precompiler and Forms applications), you might want to have application administrators. An application administrator is responsible for the following types of tasks:

- creating roles for an application and managing the privileges of each application role
- creating and managing the objects used by a database application
- maintaining and updating the application code and Oracle procedures and packages, as necessary

Often, an application administrator is also the application developer that designed the application. However, these jobs might not be the responsibility of the developer, and can be assigned to another individual familiar with the database application.

Auditing Policy

Security administrators should define a policy for the auditing procedures of each database. You may, for example, decide to have database auditing disabled unless questionable activities are suspected. When auditing is required, the security administrator must decide what level of detail to audit the database; usually, general system auditing is followed by more specific types of auditing after the origins of suspicious activity are determined.

CHAPTER

19

Managing Users and Resources

This chapter describes how to control access to an Oracle database, and includes the following topics:

- Session and User Licensing
- User Authentication
- Oracle Users
- Managing Resources with Profiles
- Listing Information About Database Users and Profiles

See Also: For guidelines on establishing security policies for users and profiles, see Chapter 18.

Privileges and roles control the access a user has to a database and the schema objects within the database. For information on privileges and roles, see Chapter 20.

For databases using Trusted Oracle, see the *Trusted Oracle7 Server Administrator's Guide* for additional information about user management in that environment.

This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager/GUI or Server Manager/LineMode, see the *Oracle Server Manager User's Guide*.

Session and User Licensing

This section describes aspects of session and user licensing, and includes the following topics:

- Concurrent Usage Licensing
- Connecting Privileges
- Named User Licensing
- Viewing Licensing Limits and Current Values

Oracle helps you ensure that your site complies with its Oracle Server license agreement. If your site is licensed by concurrent usage, you can track and limit the number of sessions concurrently connected to a database. If your site is licensed by named users, you can limit the number of named users created in a database. In either case, you control the licensing facilities, and must enable the facilities and set the appropriate limits.

To use the licensing facility, you need to know which type of licensing agreement your site has, and what the maximum number of sessions or named users is. Your site may use either type of licensing (concurrent usage or named user), but not both.

> **Note:** In a few cases, a site might have an unlimited license, rather than concurrent usage or named user licensing. In these cases only, leave the licensing mechanism disabled, and omit LICENSE_MAX_SESSIONS, LICENSE_SESSIONS_WARNING, and LICENSE_MAX_USERS from the parameter file, or set the value of all three to 0.

Concurrent Usage Licensing

Concurrent usage licensing limits the number of sessions that can be connected simultaneously to the database on the specified computer. You can set a limit on the number of concurrent sessions before you start an instance. In fact, you should have set this limit as part of the initial installation procedure. You can also change the maximum number of concurrent sessions while the database is running.

See Also: For information about the initial installation procedure, see “Edit New Parameter Files” on page 2 – 5.

Connecting Privileges

After your instance’s session limit is reached, only users with RESTRICTED SESSION privilege (usually DBAs) can connect to the database. When a user with RESTRICTED SESSION privileges connects, Oracle sends the user a message indicating that the maximum limit has been reached, and writes a message to the ALERT file. When the maximum is reached, you should connect only to terminate unneeded processes. Do not raise the licensing limits unless you have upgraded your Oracle license agreement.

In addition to setting a maximum concurrent session limit, you can set a warning limit on the number of concurrent sessions. After this limit is reached, additional users can continue to connect (up to the maximum limit); however, Oracle writes an appropriate message to the ALERT file with each connection, and sends each connecting user who has the RESTRICTED SESSION privilege a warning indicating that the maximum is about to be reached.

If a user is connecting with administrator privileges, the limits still apply; however, Oracle enforces the limit after the first statement the user executes.

In addition to enforcing the concurrent usage limits, Oracle tracks the highest number of concurrent sessions for each instance. You can use this “high water mark.”

See Also: For information about terminating sessions, see “Terminating Sessions” on page 4 – 16.

For information about Oracle licensing limit upgrades, see “Viewing Licensing Limits and Current Values” on page 19 – 5.

Parallel Server Concurrent Usage Limits

For instances running with the Parallel Server, each instance can have its own concurrent usage limit and warning limit. However, the sum of the instances’ limits must not exceed the site’s concurrent usage license.

Warning: Sessions that connect to Oracle through multiplexing software or hardware (such as a TP monitor) each contribute individually to the concurrent usage limit. However, the Oracle licensing mechanism cannot distinguish the number of sessions connected this way. If your site uses multiplexing software or hardware, you must consider that and set the maximum concurrent usage limit lower to account for the multiplexed sessions.

See Also: For more information about setting and changing limits in a parallel server environment, see the *Oracle7 Parallel Server Concepts & Administration* guide.

Setting the Maximum Number of Sessions

To set the maximum number of concurrent sessions for an instance, set the parameter LICENSE_MAX_SESSIONS as follows:

```
LICENSE_MAX_SESSIONS = 80
```

If you set this limit, you are not required to set a warning limit (LICENSE_SESSIONS_WARNING). However, using the warning limit makes the maximum limit easier to manage, because it gives you advance notice that your site is nearing maximum use.

Setting the Session Warning Limit

To set the warning limit for an instance, set the parameter LICENSE_SESSIONS_WARNING in the parameter file used to start the instance.

Set the session warning to a value lower than the concurrent usage maximum limit (LICENSE_MAX_SESSIONS).

Changing Concurrent Usage Limits While the Database is Running

To change either the maximum concurrent usage limit or the warning limit while the database is running, use the ALTER SYSTEM command with the appropriate option. The following statement changes the maximum limit to 100 concurrent sessions:

```
ALTER SYSTEM SET LICENSE_MAX_SESSIONS = 100;
```

The following statement changes both the warning limit and the maximum limit:

```
ALTER SYSTEM
   SET LICENSE_MAX_SESSIONS = 64
      LICENSE_SESSIONS_WARNING = 54;
```

If you change either limit to a value lower than the current number of sessions, the current sessions remain; however, the new limit is enforced for all future connections until the instance is shut down. To change the limit permanently, change the value of the appropriate parameter in the parameter file.

To change the concurrent usage limits while the database is running, you must have the ALTER SYSTEM privilege. Also, to connect to an instance after the instance's maximum limit has been reached, you must have the RESTRICTED SESSION privilege.

⚠ **Warning:** Do not raise the concurrent usage limits unless you have appropriately upgraded your Oracle Server license. Contact your Oracle representative for more information.

Named User Limits

Named user licensing limits the number of individuals authorized to use Oracle on the specified computer. To enforce this license, you can set a limit on the number of users created in the database before you start an instance. You can also change the maximum number of users while the instance is running, or disable the limit altogether. You cannot create more users after reaching this limit. If you try to do so, Oracle returns an error indicating that the maximum number of users have been created, and writes a message to the ALERT file.

This mechanism operates on the assumption that each person accessing the database has a unique username, and that there are no shared usernames. Do not allow multiple users to connect using the same username.

See Also: For instances running with the Parallel Server, all instances connected to the same database should have the same named user limit. See the *Oracle7 Parallel Server Concepts & Administration* guide for more information.

Setting User Limits

To limit the number of users created in a database, set the LICENSE_MAX_USERS parameter in the database's parameter file. The following example sets the maximum number of users to 200:

```
LICENSE_MAX_USERS = 200
```

If the database contains more than LICENSE_MAX_USERS when you start it, Oracle returns a warning and writes an appropriate message in the ALERT file. You cannot create additional users until the number of users drops below the limit, you should delete users, or upgrade your Oracle license.

Changing User Limits

To change the maximum named users limit, use the ALTER SYSTEM command with the LICENSE_MAX_USERS option. The following statement changes the maximum number of defined users to 300:

```
ALTER SYSTEM SET LICENSE_MAX_USERS = 300;
```

If you try to change the limit to a value lower than the current number of users, Oracle returns an error and continues to use the old limit. If you successfully change the limit, the new limit remains in effect until you shut down the instance; to change the limit permanently, change the value of LICENSE_MAX_USERS in the parameter file.

To change the maximum named users limit, you must have the ALTER SYSTEM privilege.

⚠ **Warning:** Do not raise the named user limit unless you have appropriately upgraded your Oracle license. Contact your Oracle representative for more information.

Viewing Licensing Limits and Current Values

You can see the current limits of all of the license settings, the current number of sessions, and the maximum number of concurrent sessions for the instance by querying the V$LICENSE data dictionary view. You can use this information to determine if you need to upgrade your Oracle license to allow more concurrent sessions or named users:

```
SELECT sessions_max s_max,
   sessions_warning s_warning,
   sessions_current s_current,
   sessions_highwater s_high,
   users_max
   FROM v$license;

S_MAX      S_WARNING  S_CURRENT  S_HIGH     USERS_MAX
-------------------------------------------------------
     100          80         65         88        250
```

In addition, Oracle writes the session high water mark to the database's ALERT file when the database shuts down, so you can check for it there.

To see the current number of named users defined in the database, use the following query:

```
SELECT COUNT(*) FROM dba_users;

COUNT(*)
----------
       174
```

User Authentication

This section describes aspects of authenticating users, and includes the following topics:

- Operating System Authentication
- Database Authentication

Depending on the way you want user identities to be authenticated before they are allowed to create a database session, there are two ways to define users.

1. You can configure Oracle so that it performs both identification and authentication of users.
2. You can configure Oracle so that it performs only the identification of users (leaving authentication up to the operating system or network security service).

You can use the following statement to create a user who is identified and authenticated by Oracle:

```
CREATE USER scott IDENTIFIED BY tiger;
```

Use the following command to create a user who is identified by Oracle and authenticated by the operating system or a network service:

```
CREATE USER scott IDENTIFIED EXTERNALLY;
```

Using CREATE USER IDENTIFIED EXTERNALLY, database administrators can create database accounts that must be authenticated via the operating system and cannot be authenticated using a password. By default, Oracle only allows operating system authenticated logins over secure connections. Setting the initialization parameter REMOTE_OS_AUTHENT to TRUE allows the RDBMS to trust the client's operating system username received over a non–secure connection and use it for account access. When creating a user, you can specify how that user is authenticated by Oracle. Of course, you can alter any user authentication methods later, after creating a user.

See Also: For information about network authentication, see *Oracle7 Server Distributed Systems, Volume I.*

Operating System Authentication

With operating system authentication your database relies on the underlying operating system to restrict access to database accounts. A database password is not used for this type of login. If your operating system permits, you can have it authenticate users. If you do so, set the parameter OS_AUTHENT_PREFIX, and use this prefix in Oracle usernames. This parameter defines a prefix that Oracle adds to the beginning of every user's operating system account name. Oracle compares the prefixed username with the Oracle usernames in the database when a user attempts to connect.

For example, assume that OS_AUTHENT_PREFIX is set as follows:

```
OS_AUTHENT_PREFIX=OPS$
```

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

The default value of this parameter is "OPS$" for backward compatibility with previous versions of Oracle. However, you might prefer to set the prefix value to some other string or a null string (an empty set of double quotes: ""). Using a null string eliminates the addition of any prefix to operating system account names, so that Oracle usernames exactly match operating system usernames.

After you set OS_AUTHENT_PREFIX, it should remain the same for the life of a database. If you change the prefix, any database username that includes the old prefix cannot be used to establish a connection, unless you alter the username to have it use password authentication.

See Also: The text of the OS_AUTHENT_PREFIX parameter is case–sensitive on some operating systems. See your operating system–specific Oracle documentation for more information about this initialization parameter.

Operating System Authentication and Network Clients

If you want to have the operating system authenticate a user, by default that user cannot connect to the database over SQL*Net. This means the user cannot connect using a multi–threaded server, as this connection uses SQL*Net. This restriction is the default because a remote user could impersonate another operating system user over a network connection.

If you are not concerned with this security risk and want to use operating system user authentication with network clients, set the parameter REMOTE_OS_AUTHENT (default is FALSE)in the database's parameter file to TRUE. The change will take effect the next time you start the instance and mount the database.

Database Authentication

To have Oracle authenticate a user, specify a password for the user when you create or alter the user. Users can change their password at any time. Passwords are stored in an encrypted format. Each password must be made up of single–byte characters, even if your database uses a multi–byte character set.

See Also: For more information about valid passwords, see the *Oracle7 Server SQL Reference.*

Oracle Users

Each Oracle database has a list of valid database users. To access a database, a user must run a database application and connect to the database instance using a valid username defined in the database. This section explains how to manage users for a database, and includes the following topics:

- Creating Users
- Altering Users
- Dropping Users

Creating Users

To create a database user, you must have the CREATE USER system privilege. When creating a new user, tablespace quotas can be specified for any tablespace in the database, even if the creator does not have a quota on a specified tablespace. Due to such privileged power, a security administrator is normally the only type of user that has the CREATE USER system privilege.

You create a user with either the Create User property sheet of Server Manager/GUI, or the SQL command CREATE USER. Using either option, you can also specify the new user's default and temporary segment tablespaces, tablespace quotas, and profile.

The following statement creates a new user named JWARD, identified externally:

```
CREATE USER OPS$jward
   IDENTIFIED EXTERNALLY
   DEFAULT TABLESPACE data_ts
   TEMPORARY TABLESPACE temp_ts
   QUOTA 100M ON test_ts
   QUOTA 500K ON data_ts
   PROFILE clerk;
```

See Also: A newly–created user cannot connect to the database until granted the CREATE SESSION system privilege; see page 20 – 12.

Specifying a Name

Within each database, a username must be unique with respect to other usernames and roles; a user and role cannot have the same name. Furthermore, each user has an associated schema. Within a schema, each schema object must have unique names.

Usernames in Multi–Byte Character Sets In a database that uses a multi–byte character set, each username should contain at least one single–byte character. If a username contains only multi–byte characters, the encrypted username/password combination is considerably less secure.

Setting a User's Authentication

In the previous CREATE USER statement, the new user is to be authenticated using the operating system. The username includes the default prefix "OPS$." If the OS_AUTHENT_PREFIX parameter is set differently (that is, if it specifies either no prefix or some other prefix), modify the username accordingly, by omitting the prefix or substituting the correct prefix.

Alternatively, you can create a user who is authenticated using the database and a password:

```
CREATE USER jward
   IDENTIFIED BY airplane
   . . . ;
```

In this case, the connecting user must supply the correct password to the database to connect successfully.

User Passwords in Multi–Byte Character Sets In a database that uses a multi–byte character set, passwords must include only single–byte characters. Multi–byte characters are not accepted in passwords.

See Also: For more information about valid passwords, see the *Oracle7 Server SQL Reference*.

Assigning a Default Tablespace

Each user has a default tablespace. When a user creates a schema object and specifies no tablespace to contain it, Oracle stores the object in the user's default tablespace.

The default setting for every user's default tablespace is the SYSTEM tablespace. If a user does not create objects, this default setting is fine. However, if a user creates any type of object, consider specifically setting the user's default tablespace. You can set a user's default tablespace during user creation, and change it later. Changing the user's default tablespace affects only objects created after the setting is changed.

Consider the following issues when deciding which tablespace to specify:

- Set a user's default tablespace only if the user has the privileges to create objects (such as tables, views, and clusters).
- Set a user's default tablespace to a tablespace for which the user has a quota.
- If possible, set a user's default tablespace to a tablespace other than the SYSTEM tablespace to reduce contention between data dictionary objects and user objects for the same datafiles.

In the previous CREATE USER statement, JWARD's default tablespace is DATA_TS.

Assigning a Temporary Tablespace

Each user also has a temporary tablespace. When a user executes a SQL statement that requires a temporary segment, Oracle stores the segment in the user's temporary tablespace.

If a user's temporary tablespace is not explicitly set, the default is the SYSTEM tablespace. However, setting each user's temporary tablespace reduces file contention among temporary segments and other types of segments. You can set a user’s temporary tablespace at user creation, and change it later.

In the previous CREATE USER statement, JWARD’s temporary tablespace is TEMP_TS, a tablespace created explicitly to only contain temporary segments.

Assigning Tablespace Quotas

You can assign each user a tablespace quota for any tablespace. Assigning a quota does two things:

- Users with privileges to create certain types of objects can create those objects in the specified tablespace.
- Oracle limits the amount of space that can be allocated for storage of a user’s objects within the specified tablespace to the amount of the quota.

By default, a user has no quota on any tablespace in the database. If the user has the privilege to create a schema object, you must assign a quota to allow the user to create objects. Minimally, assign users a quota for the default tablespace, and additional quotas for other tablespaces in which they will create objects.

You can assign a user either individual quotas for a specific amount of disk space in each tablespace, or an unlimited amount of disk space in all tablespaces. Specific quotas prevent a user’s objects from consuming too much space in the database.

You can assign a user’s tablespace quotas when you create the user, or add or change quotas later. If a new quota is less than the old one, then the following conditions hold true:

- If a user has already exceeded a new tablespace quota, the user’s objects in the tablespace cannot be allocated more space until the combined space of these objects falls below the new quota.
- If a user has not exceeded a new tablespace quota, or if the space used by the user’s objects in the tablespace falls under a new tablespace quota, the user’s objects can be allocated space up to the new quota.

Revoking Tablespace Access You can revoke a user’s tablespace access by changing the user’s current quota to zero. After a quota of zero is assigned, the user’s objects in the revoked tablespace remain, but the objects cannot be allocated any new space.

UNLIMITED TABLESPACE System Privilege To permit a user to use an unlimited amount of any tablespace in the database, grant the user the UNLIMITED TABLESPACE system privilege. This overrides all explicit tablespace quotas for the user. If you later revoke the privilege, explicit quotas again take effect. You can grant this privilege only to users, not to roles.

Before granting the UNLIMITED TABLESPACE system privilege, consider the consequences of doing so:

Advantage

- You can grant a user unlimited access to all tablespaces of a database with one statement.

Disadvantages

- The privilege overrides all explicit tablespace quotas for the user.
- You cannot selectively revoke tablespace access from a user with the UNLIMITED TABLESPACE privilege. You can grant access selectively only after revoking the privilege.

Setting Default Roles

You cannot set a user's default roles in the CREATE USER statement. When you first create a user, the user's default role setting is ALL, which causes all roles subsequently granted to the user to be default roles. Use the ALTER USER command to change the user's default roles.

Warning: When you create a role, it is granted to you implicitly and added as a default role. You will get an error at login if you have more than MAX_ENABLED_ROLES. You can avoid this error by altering the user's default roles to be less than MAX_ENABLED_ROLES. Thus, you should change the DEFAULT ROLE settings of SYS and SYSTEM before creating user roles.

Altering Users

Users can change their own passwords. However, to change any other option of a user's security domain, you must have the ALTER USER system privilege. Security administrators are normally the only users that have this system privilege, as it allows a modification of *any* user's security domain. This privilege includes the ability to set tablespace quotas for a user on any tablespace in the database, even if the user performing the modification does not have a quota for a specified tablespace.

You can alter a user's security settings with either the Alter User property sheet of Server Manager/GUI, or the SQL command ALTER USER. Changing a user's security settings affects the user's future sessions, not current sessions.

The following statement alters the security settings for user AVYRROS:

```
ALTER USER avyrros
   IDENTIFIED EXTERNALLY
   DEFAULT TABLESPACE data_ts
   TEMPORARY TABLESPACE temp_ts
   QUOTA 100M ON data_ts
   QUOTA 0 ON test_ts
   PROFILE clerk;
```

The ALTER USER statement here changes AVYRROS's security settings as follows:

- Authentication is changed to use AVYRROS's operating system account.
- AVYRROS's default and temporary tablespaces are explicitly set.
- AVYRROS is given a 100M quota for the DATA_TS tablespace.
- AVYRROS's quota on the TEST_TS is revoked.
- AVYRROS is assigned the CLERK profile.

Changing a User's Password, for Non–DBAs

While most non–DBA users do not use Server Manager, they can still change their own passwords with the ALTER USER command, as follows:

```
ALTER USER andy
   IDENTIFIED BY swordfish;
```

Users can change their own passwords this way, without any special privileges (other than those to connect to the database). Users should be encouraged to change their passwords frequently.

A user must have the ALTER USER privilege to change between Oracle authorization and operating system authorization; usually only DBAs should have this privilege.

Passwords in Multi–Byte Character Sets In a database that uses a multi–byte character set, passwords must include only single–byte characters. Multi–byte characters are not accepted in passwords.

See Also: For more information about valid passwords, see the *Oracle7 Server SQL Reference.*

Changing a User's Default Roles

A default role is one that is automatically enabled for a user when the user creates a session. You can assign a user zero or more default roles. Any role directly granted to a user can potentially be a default role of the user; you cannot specify an indirectly granted role when listing default roles in an ALTER USER DEFAULT ROLE command. However, if the role that the indirectly granted role is granted to is a default role, then all indirectly granted roles of that role are enabled by default. The number of default roles for a user should not exceed the maximum number of enabled roles that are allowed per user; if it does, when the user tries to connect, errors are returned and the connection is not allowed.

> **Note:** Oracle automatically enables a user's default roles when the user creates a session. Placing a role in a user's list of default roles bypasses authentication for the role, whether the role is defined to be authorized using a password or the operating system.

If you specify a list of roles, all other roles granted to that user are removed from the user's default role list.

Suppose user AVYRROS has been granted the roles DEVELOPER and CLERK, and CLERK is his only default role. The following statement removes CLERK from his default role list and adds DEVELOPER:

```
ALTER USER avyrros
   DEFAULT ROLE DEVELOPER;
```

In this case, any roles subsequently granted to AVYRROS will not be default roles, and will be disabled on connection.

If you specify ALL for the user's list of default roles, every role granted directly to the user is automatically added to the user's list of default roles. Subsequent modification of a user's default role list can remove newly granted roles from a user's list of default roles. The following example causes all roles currently granted to AVYRROS to be added to his list of default roles, as well as all roles granted in the future:

```
ALTER USER avyrros
   DEFAULT ROLE ALL;
```

Furthermore, you can specify ALL EXCEPT with a list of roles, and those roles will be the only roles granted to the user not on the default role list. For example, the following statement adds all roles currently granted to AVYRROS (except the role PAYROLL) to the user's default role list. Any roles granted to AVYRROS in the future are also added to the default role list:

```
ALTER USER avyrros
   DEFAULT ROLE ALL EXCEPT payroll;
```

To ensure a user has no default roles, specify NONE for the user's list of default roles:

```
ALTER USER avyrros
   DEFAULT ROLE NONE;
```

Changing a user’s default role list affects subsequent sessions; it does not affect any session in progress at the time.

Revoking a role from a user automatically removes the role from the user’s default role list.

Dropping Users

When a user is dropped, the user and associated schema is removed from the data dictionary and all schema objects contained in the user’s schema, if any, are immediately dropped.

> **Note:** If a user’s schema and associated objects must remain but the user must be revoked access to the database, revoke the CREATE SESSION privilege from the user.

A user that is currently connected to a database cannot be dropped. To drop a connected user, you must first terminate the user’s sessions using either Server Manager/GUI, or the SQL command ALTER SYSTEM with the KILL SESSION clause.

To drop a user and all the user’s schema objects (if any), you must have the DROP USER system privilege. Because the DROP USER system privilege is so powerful, a security administrator is typically the only type of user that has this privilege.

You can drop a user from a database using either the Drop menu item of Server Manager/GUI, or the SQL command DROP USER.

If the user’s schema contains any schema objects, use the CASCADE option to drop the user and all associated objects and foreign keys that depend on the tables of the user successfully. If you do not specify CASCADE and the user’s schema contains objects, an error message is returned and the user is not dropped. Before dropping a user whose schema contains objects, thoroughly investigate which objects the user’s schema contains and the implications of dropping them before the user is dropped. Pay attention to any unknown cascading effects. For example, if you intend to drop a user who owns a table, check whether any views or procedures depend on that particular table.

The following statement drops the user JONES, all objects in JONES’ schema, and any dependent foreign keys:

```
DROP USER jones CASCADE;
```

See Also: For more information about terminating sessions, see “Terminating Sessions” on page 4 – 16.

Managing Resources with Profiles

A profile is a named set of resource limits. If resource limits are turned on, Oracle limits database usage and instance resources to whatever is defined in the user's profile. You can assign a profile to each user, and a default profile to all users who do not have specific profiles. For profiles to take effect, resource limits must be turned on for the database as a whole.

This section describes aspects of profile management, and includes the following topics:

- Creating Profiles
- Assigning Profiles
- Altering Profiles
- Using Composite Limits
- Dropping Profiles
- Enabling and Disabling Resource Limits

Creating Profiles

To create a profile, you must have the CREATE PROFILE system privilege. You can create profiles using either the Create Profile property sheet of Server Manager / GUI, or the SQL command CREATE PROFILE. At the same time, you can explicitly set particular resource limits.

The following statement creates the profile CLERK:

```
CREATE PROFILE clerk LIMIT
   SESSIONS_PER_USER 2
   CPU_PER_SESSION unlimited
   CPU_PER_CALL 6000
   LOGICAL_READS_PER_SESSION unlimited
   LOGICAL_READS_PER_CALL 100
   IDLE_TIME 30
   CONNECT_TIME 480;
```

All unspecified resource limits for a new profile take the limit set by the DEFAULT profile. You can also specify limits for the DEFAULT profile.

Using the DEFAULT Profile

Each database has a DEFAULT profile, and its limits are used in two cases:

- If a user is not explicitly assigned a profile, then the user conforms to *all* the limits of the DEFAULT profile.
- All unspecified limits of any profile use the corresponding limit of the DEFAULT profile.

Initially, all limits of the DEFAULT profile are set to UNLIMITED. However, to prevent unlimited resource consumption by users of the DEFAULT profile, the security administrator should change the default limits using the Alter Profile dialog box of Server Manager, or a typical ALTER PROFILE statement:

```
ALTER PROFILE default LIMIT
   . . . ;
```

Any user with the ALTER PROFILE system privilege can adjust the limits in the DEFAULT profile. The DEFAULT profile cannot be dropped.

Assigning Profiles

After a profile has been created, you can assign it to database users. Each user can be assigned only one profile at any given time. If a profile is assigned to a user who already has a profile, the new profile assignment overrides the previously assigned profile. Profile assignments do not affect current sessions. Profiles can be assigned only to users and not to roles or other profiles.

Profiles can be assigned to users using the Assign Profile dialog box of Server Manager / GUI, or the SQL commands CREATE USER or ALTER USER.

See Also: For more information about assigning a profile to a user, see page 19 – 9 and page 19 – 12.

Altering Profiles

You can alter the resource limit settings of any profile using either the Alter Profile property sheet of Server Manager / GUI, or the SQL command ALTER PROFILE. To alter a profile, you must have the ALTER PROFILE system privilege.

Any adjusted profile limit overrides the previous setting for that profile limit. By adjusting a limit with a value of DEFAULT, the resource limit reverts to the default limit set for the database. All profiles not adjusted when altering a profile retain the previous settings. Any changes to a profile do not affect current sessions. New profile settings are used only for sessions created after a profile is modified.

The following statement alters the CLERK profile:

```
ALTER PROFILE clerk LIMIT
   CPU_PER_CALL default
   LOGICAL_READS_PER_SESSION 20000;
```

See Also: For information about default profiles, see “Using the Default Profile” on page 19 – 16.

Using Composite Limits

You can limit the total resource cost for a session via composite limits. In addition to setting specific resource limits explicitly for a profile, you can set a single composite limit that accounts for all resource limits in a profile. You can set a profile's composite limit using the Composite Limit checkbox of the Create Profile and Alter Profile property sheets of Server Manager / GUI, or the COMPOSITE_LIMIT parameter of the SQL commands CREATE PROFILE or ALTER PROFILE. A composite limit is set via a *service unit*, which is a weighted sum of all resources used.

The following CREATE PROFILE statement is defined using the COMPOSITE_LIMIT parameter:

```
CREATE PROFILE clerk LIMIT
   COMPOSITE_LIMIT 20000
   SESSIONS_PER_USER 2
   CPU_PER_CALL 1000;
```

Notice that both explicit resource limits and a composite limit can exist concurrently for a profile. The limit that is reached first stops the activity in a session. Composite limits allow additional flexibility when limiting the use of system resources.

Determining the Value of the Composite Limit

The correct service unit setting for a composite limit depends on the total amount of resource used by an average profile user. As with each specific resource limit, historical information should be gathered to determine the normal range of composite resource usage for a typical profile user.

Setting Resource Costs

Each system has its own characteristics; some system resources may be more valuable than others. Oracle enables you to give each system resource a *cost*. Costs weight each system resource at the database level. Costs are only applied to the composite limit of a profile; costs do not apply to set individual resource limits explicitly.

To set resource costs, you must have the ALTER RESOURCE system privilege.

Only certain resources can be given a cost, including CPU_PER_SESSION, LOGICAL_READS_PER_SESSION, CONNECT_TIME, and PRIVATE_SGA. Set costs for a database using the SQL command ALTER RESOURCE COST:

```
ALTER RESOURCE COST
   CPU_PER_SESSION 1
   LOGICAL_READS_PER_SESSION 50;
```

A large cost means that the resource is very expensive, while a small cost means that the resource is not expensive. By default, each resource is initially given a cost of 0. A cost of 0 means that the resource should not be considered in the composite limit (that is, it does not cost anything to use this resource). No resource can be given a cost of NULL.

See Also: For additional information and recommendations on setting resource costs, see your operating system–specific Oracle documentation.

Dropping Profiles

To drop a profile, you must have the DROP PROFILE system privilege. You can drop a profile using either Server Manager / GUI, or the SQL command DROP PROFILE. To successfully drop a profile currently assigned to a user, use the CASCADE option.

The following statement drops the profile CLERK, even though it is assigned to a user:

```
DROP PROFILE clerk CASCADE;
```

Any user currently assigned to a profile that is dropped is automatically assigned to the DEFAULT profile. The DEFAULT profile cannot be dropped. Note that when a profile is dropped, the drop does not affect currently active sessions; only sessions created after a profile is dropped abide by any modified profile assignments.

Enabling and Disabling Resource Limits

A profile can be created, assigned to users, altered, and dropped at any time by any authorized database user, but the resource limits set for a profile are enforced only when you enable resource limitation for the associated database. Resource limitation enforcement can be enabled or disabled by two different methods, as described in the next two sections.

To alter the enforcement of resource limitation while the database remains open, you must have the ALTER SYSTEM system privilege.

Enabling and Disabling Resource Limits Before Startup

If a database can be temporarily shut down, resource limitation can be enabled or disabled by the RESOURCE_LIMIT initialization parameter in the database's parameter file. Valid values for the parameter are TRUE (enables enforcement) and FALSE; by default, this parameter's value is set to FALSE. Once the parameter file has been edited, the database instance must be restarted to take effect. Every time an instance is started, the new parameter value enables or disables the enforcement of resource limitation.

Enabling and Disabling Resource Limits While the Database is Open

If a database cannot be temporarily shut down or the resource limitation feature must be altered temporarily, you can enable or disable the enforcement of resource limitation using the SQL command ALTER SYSTEM. After an instance is started, an ALTER SYSTEM statement overrides the value set by the RESOURCE_LIMIT parameter. For example, the following statement enables the enforcement of resource limitation for a database:

```
ALTER SYSTEM
   SET RESOURCE_LIMIT = TRUE;
```

An ALTER SYSTEM statement does not permanently determine the enforcement of resource limitation. If the database is shut down and restarted, the enforcement of resource limits is determined by the value set for the RESOURCE_LIMIT parameter.

Listing Information About Database Users and Profiles

The data dictionary stores information about every user and profile, including the following:

- all users in a database
- each user's default tablespace for tables, clusters, and indexes
- each user's tablespace for temporary segments
- each user's space quotas, if any
- each user's assigned profile and resource limits
- the cost assigned to each applicable system resource
- each current session's memory usage

The following data dictionary views may be of interest when you work with database users and profiles:

- ALL_USERS
- USER_USERS
- DBA_USERS
- USER_TS_QUOTAS
- DBA_TS_QUOTAS
- USER_RESOURCE_LIMITS
- DBA_PROFILES
- RESOURCE_COST
- V$SESSION
- V$SESSTAT
- V$STATNAME

See Also: See the *Oracle7 Server Reference* for detailed information about each view.

Listing Information about Users and Profiles: Examples

The examples in this section assume a database in which the following statements have been executed:

```
CREATE PROFILE clerk LIMIT
   SESSIONS_PER_USER 1
   IDLE_TIME 30
   CONNECT_TIME 600;

CREATE USER jfee
   IDENTIFIED BY wildcat
   DEFAULT TABLESPACE users
   TEMPORARY TABLESPACE temp_ts
   QUOTA 500K ON users
   PROFILE clerk;

CREATE USER tsmith
   IDENTIFIED BY bedrock
   DEFAULT TABLESPACE users
   TEMPORARY TABLESPACE temp_ts
   QUOTA unlimited ON users;
```

Listing All Users and Associated Information

The following query lists all users defined in the database:

```
SELECT * FROM sys.dba_users;

USERNA  USER_ID PASSWORD         DEFAUL TEMPOR  CREATED   PROFILE
------ -------- ---------------- ------ ------  --------- --------
SYS          % 522D06CDE017CF93 SYSTEM SYSTEM  31-JUL-90 PUBLIC...
SYSTEM       % 9B30B3EB7A7EE46A SYSTEM SYSTEM  31-JUL-90 PUBLIC...
JFEE         % DEE4F647381D62C4 USERS  TEMP_TS 12-SEP-90 CLERK
TSMITH       % 4791F162172E7834 USERS  TEMP_TS 12-SEP-90 PUBLIC...
```

All passwords are encrypted to preserve security.

Listing Users' Roles

The following query lists, for each user, the roles granted to that user, and indicates whether each role is granted with the ADMIN OPTION and is a default role:

```
SELECT * FROM sys.dba_role_privs where grantee = 'JFEE';

GRANTEE                    GRANTED_ROLE              ADM DEF
-------------------------- ------------------------- --- ---
JFEE                       CLERK                     YES YES
JFEE                       PAYROLL                   NO  NO
JFEE                       WEEKLY_ADMIN              NO  NO
```

Listing All Tablespace Quotas

The following query lists all tablespace quotas specifically assigned to each user:

```
SELECT * FROM sys.dba_ts_quotas;

TABLESPACE USERNAME        BYTES  MAX_BYTES     BLOCKS MAX_BLOCKS
---------- ---------- ---------- ---------- ---------- ----------
SYSTEM     SYSTEM              0          0          0          0
SYSTEM     JFEE                0     512000          0        250
SYSTEM     TSMITH              0         –1          0         –1
```

When specific quotas are assigned, the exact number is indicated in the MAX_BYTES column. Unlimited quotas are indicated by “–1”.

Listing All Profiles and Assigned Limits

The following query lists all profiles in the database and associated settings for each limit in each profile:

```
SELECT * FROM sys.dba_profiles
   ORDER BY profile;

PROFILE          RESOURCE_NAME                    LIMIT
---------------  -------------------------------- --------------
CLERK            COMPOSITE_LIMIT                  UNLIMITED
CLERK            SESSIONS_PER_USER                1
CLERK            CPU_PER_SESSION                  UNLIMITED
CLERK            CPU_PER_CALL                     UNLIMITED
CLERK            LOGICAL_READS_PER_SESSION        UNLIMITED
CLERK            LOGICAL_READS_PER_CALL           UNLIMITED
CLERK            IDLE_TIME                        30
CLERK            CONNECT_TIME                     600
CLERK            PRIVATE_SGA                      UNLIMITED
DEFAULT          COMPOSITE_LIMIT                  UNLIMITED
DEFAULT          SESSIONS_PER_USER                UNLIMITED
DEFAULT          CPU_PER_SESSION                  UNLIMITED
DEFAULT          CPU_PER_CALL                     UNLIMITED
DEFAULT          LOGICAL_READS_PER_SESSION        UNLIMITED
DEFAULT          LOGICAL_READS_PER_CALL           UNLIMITED
DEFAULT          IDLE_TIME                        UNLIMITED
DEFAULT          CONNECT_TIME                     UNLIMITED
DEFAULT          PRIVATE_SGA                      UNLIMITED
```

Viewing Memory Use Per User Session

The following query lists all current sessions, showing the Oracle user and current memory use per session:

```
SELECT username, value || ’bytes’ ”Current session memory”
   FROM v$session sess, v$sesstat stat, v$statname name
WHERE sess.sid = stat.sid
   AND stat.statistic# = name.statistic#
   AND name.name = ’session memory’;
```

The amount of space indicated in “Current session memory” is allocated in the shared pool for each session connected through the multi–threaded server. You can limit the amount of memory allocated per user with the PRIVATE_SGA resource limit.

To see the maximum memory ever allocated to each session since the instance started, replace ’session memory’ in the query above with ’max session memory’.

CHAPTER

20

Managing User Privileges and Roles

This chapter explains how to control the capability to execute system operations and access to schema objects using privileges and roles. The following topics are included:

- Identifying User Privileges
- Managing User Roles
- Granting User Privileges and Roles
- Revoking User Privileges and Roles
- Granting Roles Using the Operating System or Network
- Listing Users' Privilege and Role Information

See Also: For information about controlling access to a database, see Chapter 19.

For suggested general database security policies, see Chapter 18.

If you are using Trusted Oracle7 in DBMS MAC mode, see the *Trusted Oracle7 Server Administrator's Guide* for important information about system privileges and role management.

This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager/GUI or Server Manager/LineMode, see the *Oracle Server Manager User's Guide*.

Identifying User Privileges

This section describes Oracle user privileges, and includes the following topics:

- System Privileges
- Object Privileges

A user *privilege* is a right to execute a particular type of SQL statement, or a right to access another user's object. Oracle also provides shortcuts for grouping privileges that are commonly granted or revoked together.

System Privileges

There are over 80 distinct system privileges. Each system privilege allows a user to perform a particular database operation or class of database operations. Table 20 – 1 lists all system privileges and the operations that they permit.

Warning: System privileges are very powerful, and should be cautiously granted to roles and trusted users of the database.

System Privilege	Operations Permitted
ANALYZE	
ANALYZE ANY	Analyze any table, cluster, or index in the database.
AUDIT	
AUDIT ANY	Audit any schema object in the database.
AUDIT SYSTEM	Enable and disable statement and privilege audit options.
CLUSTER	
CREATE CLUSTER	Create a cluster in own schema.
CREATE ANY CLUSTER	Create a cluster in any schema. Behaves similarly to CREATE ANY TABLE.
ALTER ANY CLUSTER	Alter any cluster in the database.
DROP ANY CLUSTER	Drop any cluster in the database.
DATABASE	
ALTER DATABASE	Alter the database; add files to the operating system via Oracle, regardless of operating system privileges.
DATABASE LINK	
CREATE DATABASE LINK	Create private database links in own schema.
INDEX	
CREATE ANY INDEX	Create an index in any schema on any table.
ALTER ANY INDEX	Alter any index in the database.
DROP ANY INDEX	Drop any index in the database.

Table 20 – 1 System Privileges, continued on next page

System Privilege	Operations Permitted
PRIVILEGE	
GRANT ANY PRIVILEGE	Grant any system privilege (not object privileges).
PROCEDURE	
CREATE PROCEDURE	Create stored procedures, functions, and packages in own schema.
CREATE ANY PROCEDURE	Create stored procedures, functions, and packages in any schema. (Requires that user also have ALTER ANY TABLE, BACKUP ANY TABLE, DROP ANY TABLE, SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE, or GRANT ANY TABLE.)
ALTER ANY PROCEDURE	Compile any stored procedure, function, or package in any schema.
DROP ANY PROCEDURE	Drop any stored procedure, function, or package in any schema.
EXECUTE ANY PROCEDURE	Execute any procedure or function (stand–alone or packaged), or reference any public package variable in any schema.
PROFILE	
CREATE PROFILE	Create profiles.
ALTER PROFILE	Alter any profile in the database.
DROP PROFILE	Drop any profile in the database.
ALTER RESOURCE COST	Set costs for resources used in all user sessions.
PUBLIC DATABASE LINK	
CREATE PUBLIC DATABASE LINK	Create public database links.
DROP PUBLIC DATABASE LINK	Drop public database links.
PUBLIC SYNONYM	
CREATE PUBLIC SYNONYM	Create public synonyms.
DROP PUBLIC SYNONYM	Drop public synonyms.
ROLE	
CREATE ROLE	Create roles.
ALTER ANY ROLE	Alter any role in the database.
DROP ANY ROLE	Drop any role in the database.
GRANT ANY ROLE	Grant any role in the database.
ROLLBACK SEGMENT	
CREATE ROLLBACK SEGMENT	Create rollback segments.

Table 20 – 1 System Privileges, continued on next page

System Privilege	Operations Permitted
ALTER ROLLBACK SEGMENT	Alter rollback segments.
DROP ROLLBACK SEGMENT	Drop rollback segments.
SESSION	
CREATE SESSION	Connect to the database.
ALTER SESSION	Issue ALTER SESSION statements.
RESTRICTED SESSION	Connect when the database has been started using STARTUP RESTRICT. (The OSOPER and OSDBA roles contain this privilege.)
SEQUENCE	
CREATE SEQUENCE	Create a sequence in own schema.
CREATE ANY SEQUENCE	Create any sequence in any schema.
ALTER ANY SEQUENCE	Alter any sequence in any schema.
DROP ANY SEQUENCE	Drop any sequence in any schema.
SELECT ANY SEQUENCE	Reference any sequence in any schema.
SNAPSHOT	
CREATE SNAPSHOT	Create snapshots in own schema. (User must also have the CREATE TABLE privilege.)
CREATE SNAPSHOT	Create snapshots in any schema. (User must also have the CREATE ANY TABLE privilege.)
ALTER SNAPSHOT	Alter any snapshot in any schema.
DROP ANY SNAPSHOT	Drop any snapshot in any schema.
SYNONYM	
CREATE SYNONYM	Create a synonym in own schema.
CREATE SYNONYM	Create any synonym in any schema.
DROP ANY SYNONYM	Drop any synonym in any schema.
SYSTEM	
ALTER SYSTEM	Issue ALTER SYSTEM statements.
TABLE	
CREATE TABLE	Create tables in own schema. Also allows grantee to create indexes (including those for integrity constraints) on table in own schema. (The grantee must have a quota for the tablespace or the UNLIMITED TABLESPACE privilege.)
CREATE ANY TABLE	Create tables in any schema. (If grantee has CREATE ANY TABLE privilege and creates a table in another user's schema, the owner must have space quota on that tablespace. The table owner need not have the CREATE [ANY] TABLE privilege.)

Table 20 – 1 System Privileges, continued on next page

System Privilege	Operations Permitted
ALTER ANY TABLE	Alter any table in any schema and compile any view in any schema.
BACKUP ANY TABLE	Perform an incremental export using the Export utility of tables in any schema.
DROP ANY TABLE	Drop or truncate any table in any schema.
LOCK ANY TABLE	Lock any table or view in any schema.
COMMENT ANY TABLE	Comment on any table, view, or column in schema.
SELECT ANY TABLE	Query any table, view, or snapshot in any schema.
INSERT ANY TABLE	Insert rows into any table or view in any schema.
UPDATE ANY TABLE	Update rows in any table or view in any schema.
DELETE ANY TABLE	Delete rows from any table or view in any schema.
TABLESPACE	
CREATE TABLE SPACE	Create tablespaces; add files to the operating system via Oracle, regardless of the user's operating system privileges.
ALTER TABLESPACE	Alter tablespaces; add files to the operating system via Oracle, regardless of the user's operating system privileges.
MANAGE TABLESPACE	Take any tablespace offline, bring any tablespace online, and begin and end backups of any tablespace.
DROP TABLESPACE	Drop tablespaces.
UNLIMITED TABLESPACE	Use an unlimited amount of *any* tablespace. This privilege overrides any specific quotas assigned. If revoked, the grantee's schema objects remain but further tablespace allocation is denied unless allowed by specific tablespace quotas. *This system privilege can be granted only to users and not to roles. In general, specific tablespace quotas are assigned instead of granting this system privilege.*
TRANSACTION	
FORCE TRANSACTION	Force the commit or rollback of own in–doubt distributed transaction in the local database.
FORCE ANY TRANSACTION	Force the commit or rollback of any in–doubt distributed transaction in the local database.
TRIGGER	
CREATE TRIGGER	Create a trigger in own schema.
CREATE ANY TRIGGER	Create any trigger in any schema associated with any table in any schema.
ALTER ANY TRIGGER	Enable, disable, or compile any trigger in any schema.
DROP ANY TRIGGER	Drop any trigger in any schema.

Table 20 – 1 System Privileges, continued on next page

System Privilege	Operations Permitted
USER	
CREATE ANY USER	Create users; assign quotas on *any* tablespace, set default and temporary tablespaces, and assign a profile as part of a CREATE USER statement.
BECOME ANY USER	Become another user. (Required by any user performing a full database import.)
ALTER USER	Alter other users: change any user's password or authentication method, assign tablespace quotas, set default and temporary tablespaces, assign profiles and default roles, in an ALTER USER statement. (Not required to alter own password.)
DROP USER	Drop another user.
VIEW	
CREATE VIEW	Create a view in own schema.
CREATE ANY VIEW	Create a view in any schema. (Requires that user also have ALTER ANY TABLE, BACKUP ANY TABLE, DROP ANY TABLE, LOCK ANY TABLE, COMMENT ANY TABLE, SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE, or GRANT ANY TABLE.)
DROP ANY VIEW	Drop any view in any schema.

Table 20 – 1 System Privileges

Object Privileges

Each type of object has different privileges associated with it. Table 20 – 2 summarizes the object privileges available for each type of object.

Object Privilege	Table	View	Sequence	Procedure[1]
ALTER	✓		✓	
DELETE	✓	✓		
EXECUTE				✓
INDEX	✓[2]			
INSERT	✓	✓		
REFERENCES	✓[2]			
SELECT	✓	✓[3]	✓	
UPDATE	✓	✓		

Table 20 – 2 Object Privileges

[1] *Includes stand–alone stored procedures and functions, and public package constructs.*

[2] *Privilege cannot be granted to a role.*

[3] *Can also be granted for snapshots.*

Not all types of schema objects are included in Table 20 – 2. Many of the schema objects not listed here (such as clusters, indexes, triggers, and database links) are controlled exclusively using system privileges. For example, to alter a cluster, a user must own the cluster or have the ALTER ANY CLUSTER system privilege.

Table 20 – 3 lists the SQL statements permitted by the object privileges listed in Table 20 – 2.

Object Privilege	**SQL Statements Permitted**
ALTER	ALTER object (table or sequence)
DELETE	DELETE FROM object (table or view)
EXECUTE	EXECUTE object (procedure or function). References to public package variables
INDEX	CREATE INDEX ON object (tables only)
INSERT	INSERT INTO object (table or view)
REFERENCES	CREATE or ALTER TABLE statement defining a FOREIGN KEY integrity constraint on object (tables only)
SELECT	SELECT...FROM object (table, view, or snapshot). SQL statements using a sequence
UPDATE	UPDATE object (table or view)

Table 20 – 3 SQL Statements Permitted by Object Privileges

Object Privilege Shortcut

The ALL and ALL PRIVILEGES shortcuts grant or revoke all available object privileges for a object. This shortcut is not a privilege, rather, it is a way of granting or revoking all object privileges with one word in GRANT and REVOKE statements. Note that if all object privileges are granted using the ALL shortcut, individual privileges can still be revoked.

Likewise, all individually granted privileges can be revoked using the ALL shortcut. However, if you REVOKE ALL, and revoking causes integrity constraints to be deleted (because they depend on a REFERENCES privilege that you are revoking), you must include the CASCADE CONSTRAINTS option in the REVOKE statement.

Managing User Roles

This section describes aspects of managing roles, and includes the following topics:

- Creating a Role
- Predefined Roles

A *role* groups several privileges and roles, so that they can be granted and revoked simultaneously from users. Roles can be enabled and disabled per user.

See Also: For information about roles, see the *Oracle7 Server Concepts* manual.

Creating a Role

You can create a role using either the SQL command CREATE ROLE, or the Create Role property sheet of Server Manager.

You must have the CREATE ROLE system privilege to create a role. Typically, only security administrators have this system privilege.

> **Note:** Immediately after creation, a role has no privileges associated with it. To associate privileges with a new role, you must grant privileges or other roles to the new role.

The following statement creates the CLERK role, which is authorized by the database using the password BICENTENNIAL:

```
CREATE ROLE clerk
   IDENTIFIED BY bicentennial;
```

Role Names

You must give each role you create a unique name among existing usernames and role names of the database. Roles are not contained in the schema of any user.

Role Names in Multi–Byte Character Sets

In a database that uses a multi–byte character set, Oracle Corporation recommends that each role name contain at least one single–byte character. If a role name contains only multi–byte characters, the encrypted role name / password combination is considerably less secure.

Predefined Roles

The roles listed in Table 20 – 4 are automatically defined for Oracle databases. These roles are provided for backward compatibility to earlier versions of Oracle. You can grant and revoke privileges and roles to these predefined roles, much the way you do with any role you define.

Role Name	Privileges Granted To Role
CONNECT[1]	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE[1,2]	CREATE CLUSTER, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER
DBA[1,3,4]	All system privileges WITH ADMIN OPTION
EXP_FULL_DATABASE[5]	SELECT ANY TABLE, BACKUP ANY TABLE, INSERT, DELETE, AND UPDATE ON THE TABLES SYS.INCVID, SYS.INCFIL, AND SYS.INCEXP
IMP_FULL_DATABASE[5]	BECOME USER, WRITEDOWN[6]

Table 20 – 4 Predefined Roles

[1] *Created by SQL.BSQ.*

[2] *Grantees of the RESOURCE role also receive the UNLIMITED TABLESPACE system privilege as an explicitly grant (not as part of the RESOURCE role).*

[3] *Grantees of the DBA role also receive the UNLIMITED TABLESPACE system privilege with the ADMIN OPTION as an explicit grant (not as part of the DBA role). Therefore when the DBA role is revoked, any explicit grant of UNLIMITED TABLESPACE is also revoked.*

[4] *Also includes the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles if CATEXP.SQL has been run.*

[5] *Created by CATEXP.SQL.*

[6] *A Trusted Oracle7 privilege only; see the Trusted Oracle7 Server Administrator's Guide.*

Role Authorization

A database role can optionally require authorization when a user attempts to enable the role. Role authorization can be maintained by the database (using passwords), by the operating system, or by a network service.

To alter the authorization method for a role, you must have the ALTER ANY ROLE system privilege or have been granted the role with the ADMIN OPTION.

See Also: For more information about network roles, see *Oracle7 Server Distributed Systems, Volume I.*

Role Authorization by the Database

The use of a role can be protected by an associated password. If you are granted a role protected by a password, you can enable or disable the role only by supplying the proper password for the role in a SET ROLE command.

> **Note:** In a database that uses a multi–byte character set, passwords for roles must include only single–byte characters. Multi–byte characters are not accepted in passwords.

See Also: For more information about valid passwords, see the *Oracle7 Server Reference.*

Role Authorization by the Operating System

The following statement creates a role named ACCTS_REC and requires that the operating system authorize its use:

```
CREATE ROLE role IDENTIFIED EXTERNALLY;
```

Role authentication via the operating system is useful only when the operating system must be able to dynamically link operating system privileges with applications. When a user starts an application, the operating system grants an operating system privilege to the user. The granted operating system privilege corresponds to the role associated with the application. At this point, the application can enable the application role. When the application is terminated, the previously granted operating system privilege is revoked from the user's operating system account.

If a role is authorized by the operating system, you must configure information for each user at the operating system level. This operation is operating system–dependent.

If roles are granted by the operating system, you do not need to have the operating system authorize them also; this is redundant.

See Also: For more information about roles granted by the operating system, see page 20 – 18.

Role Authorization and Network Clients

If users connect to the database over SQL*Net, by default their roles cannot be authenticated by the operating system. This includes connections through a multi–threaded server, as this connection requires SQL*Net. This restriction is the default because a remote user could impersonate another operating system user over a network connection.

If you are not concerned with this security risk and want to use operating system role authentication for network clients, set the parameter REMOTE_OS_ROLES in the database's parameter file to TRUE. The change will take effect the next time you start the instance and mount the database. (The parameter is FALSE by default.)

Witholding Authorization

A role can also be created without authorization. If a role is created without any protection, the role can be enabled or disabled by any grantee.

Changing a Role's Authorization

You can set and change the authorization method for a role using either the Alter Role property sheet of Server Manager / GUI or the SQL command ALTER ROLE.

The following statement alters the CLERK role to be authorized externally:

```
ALTER ROLE clerk
   IDENTIFIED EXTERNALLY;
```

Changing a User's Default Roles

A user's list of default roles can be set and altered using either the Alter User dialog box of Server Manager or the SQL command ALTER USER.

See Also: See "Altering Users" on page 19 – 12 for more information about these options.

Using the ALL Keyword If the user's list of default roles is specified as ALL, every role granted to a user is automatically added to the user's list of default roles. Only subsequent modification of a user's default role list can remove newly granted roles from a user's list of default roles.

Using the MAX_ENABLED_ROLES Parameter A user can enable as many roles as specified by the initialization parameter MAX_ENABLED_ROLES. All indirectly granted roles enabled as a result of enabling a primary role are included in this count. The database administrator can alter this limitation by modifying the value for this parameter. Higher values permit each user session to have more concurrently enabled roles. However, the larger the value for this parameter, the more memory space is required on behalf of each user session; this is because the PGA size is affected for each user session, and requires four bytes per role. Determine the highest number of roles that will be concurrently enabled by any one user and use this value for the MAX_ENABLED_ROLES parameter.

Dropping Roles

In some cases, it may be applicable to drop a role from the database. The security domains of all users and roles granted a dropped role are immediately changed to reflect the absence of the dropped role's privileges. All indirectly granted roles of the dropped role are also removed from affected security domains. Dropping a role automatically removes the role from all users' default role lists.

Because the creation of objects is not dependent on the privileges received via a role, tables and other objects are not dropped when a role is dropped.

To drop a role, you must have the DROP ANY ROLE system privilege or have been granted the role with the ADMIN OPTION.

You can drop a role using either the Drop menu item of Server Manager, or the SQL command DROP ROLE.

The following statement drops the role CLERK:

```
DROP ROLE clerk;
```

Granting User Privileges and Roles

This section describes aspects of granting privileges and roles, and includes the following topics:

- Granting System Privileges and Roles
- Granting Object Privileges and Roles
- Granting Privileges on Columns

Granting System Privileges and Roles

You can grant system privileges and roles to other roles and users using either the Grant System Privileges/Roles dialog box of Server Manager, or the SQL command GRANT.

To grant a system privilege or role, you must have the ADMIN OPTION for all system privileges and roles being granted. Also, any user with the GRANT ANY ROLE system privilege can grant any role in a database.

The following statement grants the CREATE SESSION system privilege and the ACCTS_PAY role to the user JWARD:

```
GRANT create session, accts_pay
   TO jward;
```

> **Note:** Object privileges *cannot* be granted along with system privileges and roles in the same GRANT statement.

The ADMIN Option

When a user creates a role, the role is automatically granted to the creator with the ADMIN OPTION. A grantee with the ADMIN option has several expanded capabilities:

- The grantee can grant or revoke the system privilege or role to or from *any* user or other role in the database. (Users cannot revoke a role from themselves.)

- The grantee can further grant the system privilege or role with the ADMIN OPTION.
- The grantee of a role can alter or drop the role.

In the following statement, the security administrator grants the NEW_DBA role to MICHAEL:

```
GRANT new_dba TO michael WITH ADMIN OPTION;
```

The user MICHAEL cannot only use all of the privileges implicit in the NEW_DBA role, but can grant, revoke, or drop the NEW_DBA role as deemed necessary. Because of these powerful capabilities, exercise caution when granting system privileges or roles with the ADMIN OPTION. Such privileges are usually reserved for a security administrator and rarely granted to other administrators or users of the system.

Granting Object Privileges and Roles

You can grant object privileges to roles and users using the Add Privilege to Role / User dialog box of Server Manager, or the SQL command GRANT.

To grant an object privilege, you must fulfill one of the following conditions:

- You own the object specified.
- You have been granted the object privileges being granted with the GRANT OPTION.

The following statement grants the SELECT, INSERT, and DELETE object privileges for all columns of the EMP table to the users JFEE and TSMITH:

```
GRANT select, insert, delete ON emp TO jfee, tsmith;
```

To grant the INSERT object privilege for only the ENAME and JOB columns of the EMP table to the users JFEE and TSMITH, issue the following statement:

```
GRANT insert(ename, job) ON emp TO jfee, tsmith;
```

To grant all object privileges on the SALARY view to the user JFEE, use the ALL shortcut, as shown in the following example:

```
GRANT ALL ON salary TO jfee;
```

> **Note:** System privileges and roles cannot be granted along with object privileges in the same GRANT statement.

The GRANT OPTION

The user whose schema contains an object is automatically granted all associated object privileges with the GRANT OPTION. This special privilege allows the grantee several expanded privileges:

- The grantee can grant the object privilege to any user or any role in the database.
- The grantee can also grant the object privilege to other users, with or without the GRANT OPTION.
- If the grantee receives object privileges for a table with the GRANT OPTION and the grantee has the CREATE VIEW or CREATE ANY VIEW system privilege, the grantee can create views on the table and grant the corresponding privileges on the view to any user or role in the database.

The GRANT OPTION is not valid when granting an object privilege to a role. Oracle prevents the propagation of object privileges via roles so that grantees of a role cannot propagate object privileges received by means of roles.

Granting Privileges on Columns

You can grant INSERT, UPDATE, or REFERENCES privileges on individual columns in a table.

⚠ **Warning:** Before granting a column–specific INSERT privilege, determine if the table contains any columns on which NOT NULL constraints are defined. Granting selective insert capability without including the NOT NULL columns prevents the user from inserting any rows into the table. To avoid this situation, make sure that each NOT NULL column is either insertable or has a non–NULL default value. Otherwise, the grantee will not be able to insert rows into the table and will receive an error.

Grant INSERT privilege on the ACCT_NO column of the ACCOUNTS table to SCOTT:

```
GRANT INSERT (acct_no)
   ON accounts TO scott;
```

Revoking User Privileges and Roles

This section describes aspects of revoking user privileges and roles, and includes the following topics:

- Revoking System Privileges and Roles
- Revoking Object Privileges and Roles

Revoking System Privileges and Roles

You can revoke system privileges and / or roles using either the Revoke System Privileges / Roles dialog box of Server Manager, or the SQL command REVOKE.

Any user with the ADMIN OPTION for a system privilege or role can revoke the privilege or role from any other database user or role The grantor does not have to be the user that originally granted the privilege or role. Also, users with the GRANT ANY ROLE can revoke *any* role.

The following statement revokes the CREATE TABLE system privilege and the ACCTS_REC role from TSMITH:

```
REVOKE create table, accts_rec FROM tsmith;
```

> **Note:** The ADMIN OPTION for a system privilege or role cannot be selectively revoked. The privilege or role must be revoked and then the privilege or role re–granted without the ADMIN OPTION.

Revoking Object Privileges and Roles

You can revoke object privileges using Server Manager, or the SQL command REVOKE.

To revoke an object privilege, the revoker must be the original grantor of the object privilege being revoked.

For example, assuming you are the original grantor, to revoke the SELECT and INSERT privileges on the EMP table from the users JFEE and TSMITH, you would issue the following statement:

```
REVOKE select, insert ON emp
   FROM jfee, tsmith;
```

The following statement revokes all privileges (which were originally granted to the role HUMAN_RESOURCE) from the table DEPT:

```
REVOKE ALL ON dept FROM human_resources;
```

> **Note:** This statement above would only revoke the privileges that the grantor authorized, not the grants made by other users. The GRANT OPTION for an object privilege cannot be selectively revoked. The object privilege must be revoked and then re–granted without the GRANT OPTION. Users cannot revoke object privileges from themselves.

Revoking Column Selective Object Privileges

Although users can grant column selective INSERT, UPDATE, and REFERENCES privileges for tables and views, they cannot selectively revoke column specific privileges with a similar REVOKE statement. Instead, the grantor must first revoke the object privilege for all columns of a table or view, and then selectively re–grant the column specific privileges that should remain.

For example, assume that role HUMAN_RESOURCES has been granted the UPDATE privilege on the DEPTNO and DNAME columns of the table DEPT. To revoke the UPDATE privilege on just the DEPTNO column, you would issue the following two statements:

```
REVOKE UPDATE ON dept FROM human_resources;
GRANT UPDATE (dname) ON dept TO human_resources;
```

The REVOKE statement revokes UPDATE privilege on all columns of the DEPT table from the role HUMAN_RESOURCES. The GRANT statement re–grants UPDATE privilege on the DNAME column to the role HUMAN_RESOURCES.

Revoking the REFERENCES Object Privilege

If the grantee of the REFERENCES object privilege has used the privilege to create a foreign key constraint (that currently exists), the grantor can only revoke the privilege by specifying the CASCADE CONSTRAINTS option in the REVOKE statement:

```
REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;
```

Any foreign key constraints currently defined that use the revoked REFERENCES privilege are dropped when the CASCADE CONSTRAINTS options is specified.

Effects of Revoking Privileges

Depending on the type of privilege, there may be cascading effects when a privilege is revoked.

System Privileges

There are no cascading effects when revoking a system privilege related to DDL operations, regardless of whether the privilege was granted with or without the ADMIN OPTION. For example, assume the following:

1. The security administrator grants the CREATE TABLE system privilege to JFEE with the ADMIN OPTION.
2. JFEE creates a table.
3. JFEE grants the CREATE TABLE system privilege to TSMITH.
4. TSMITH creates a table.
5. The security administrator revokes the CREATE TABLE system privilege from JFEE.

6. JFEE's table continues to exist. TSMITH still has the table and the CREATE TABLE system privilege.

Cascading effects can be observed when revoking a system privilege related to a DML operation. For example, if SELECT ANY TABLE is granted to a user, and that user has created any procedures, all procedures contained in the user's schema must be re–authorized before they can be used again.

Object Privileges

Revoking an object privilege may have cascading effects that should be investigated before issuing a REVOKE statement.

- Object definitions that depend on a DML object privilege can be affected if the DML object privilege is revoked. For example, assume the procedure body of the TEST procedure includes a SQL statement that queries data from the EMP table. If the SELECT privilege on the EMP table is revoked from the owner of the TEST procedure, the procedure can no longer be executed successfully.
- Object definitions that require the ALTER and INDEX DDL object privileges are not affected if the ALTER or INDEX object privilege is revoked. For example, if the INDEX privilege is revoked from a user that created an index on someone else's table, the index continues to exist after the privilege is revoked.
- When a REFERENCES privilege for a table is revoked from a user, any foreign key integrity constraints defined by the user that require the dropped REFERENCES privilege are automatically dropped. For example, assume that the user JWARD is granted the REFERENCES privilege for the DEPTNO column of the DEPT table and creates a foreign key on the DEPTNO column in the EMP table that references the DEPTNO column. If the REFERENCES privilege on the DEPTNO column of the DEPT table is revoked, the foreign key constraint on the DEPTNO column of the EMP table is dropped in the same operation.
- The object privilege grants propagated using the GRANT OPTION are revoked if a grantor's object privilege is revoked. For example, assume that USER1 is granted the SELECT object privilege with the GRANT OPTION, and grants the SELECT privilege on EMP to USER2. Subsequently, the SELECT privilege is revoked from USER1. This revoke is cascaded to USER2 as well. Any objects that depended on USER1's and USER2's revoked SELECT privilege can also be affected, as described in previous bullet items.

Granting to and Revoking from the User Group PUBLIC

Privileges and roles can also be granted to and revoked from the user group PUBLIC. Because PUBLIC is accessible to every database user, all privileges and roles granted to PUBLIC are accessible to every database user.

Security administrators and database users should only grant a privilege or role to PUBLIC if every database user requires the privilege or role. This recommendation reinforces the general rule that at any given time, each database user should only have the privileges required to accomplish the group's current tasks successfully.

Revoking a privilege from PUBLIC can cause significant cascading effects. If any privilege related to a DML operation is revoked from PUBLIC (for example, SELECT ANY TABLE, UPDATE ON emp), all procedures in the database, including functions and packages, must be *reauthorized* before they can be used again. Therefore, exercise caution when granting DML–related privileges to PUBLIC.

See Also: For more information about object dependencies, see "Managing Object Dependencies" on page 16 – 18.

When Do Grants and Revokes Take Effect?

Depending on what is granted or revoked, a grant or revoke takes effect at different times:

- All grants / revokes of system and object privileges to anything (users, roles, and PUBLIC) are immediately observed.
- All grants / revokes of roles to anything (users, other roles, PUBLIC) are only observed when a current user session issues a SET ROLE statement to re–enable the role after the grant / revoke, or when a new user session is created after the grant / revoke.

Granting Roles Using the Operating System or Network

This section describes aspects of granting roles via your operating system or network, and includes the following topics:

- Using Operating System Role Identification
- Using Operating System Role Management
- Granting and Revoking Roles When OS_ROLES=TRUE
- Enabling and Disabling Roles When OS_ROLES=TRUE
- Using Network Connections with Operating System Role Management

Instead of a security administrator explicitly granting and revoking database roles to and from users using GRANT and REVOKE statements, the operating system that operates Oracle can grant roles to users at connect time. Roles can be administered using the operating system and passed to Oracle when a user creates a session. As part of this mechanism, each user's default roles and the roles granted to a user with the ADMIN OPTION can be identified. Even if the operating system is used to authorize users for roles, all roles must be created in the database and privileges assigned to the role with GRANT statements.

Roles can also be granted through a network service. For information about network roles, see *Oracle7 Server Distributed Systems, Volume I.*

The advantage of using the operating system to identify a user's database roles is that privilege management for an Oracle database can be externalized. The security facilities offered by the operating system control a user's privileges. This option may offer advantages of centralizing security for a number of system activities. For example, MVS Oracle administrators may want RACF groups to identify a database user's roles, UNIX Oracle administrators may want UNIX groups to identify a database user's roles, or VMS Oracle administrators may want to use rights identifiers to identify a database user's roles.

The main disadvantage of using the operating system to identify a user's database roles is that privilege management can only be performed at the role level. Individual privileges cannot be granted using the operating system, but can still be granted inside the database using GRANT statements.

A secondary disadvantage of using this feature is that by default users cannot connect to the database through the multi–threaded server, or any other network connection, if the operating system is managing roles. However, you can change this default; see "Using Network Connections with Operating System Role Management" on page 20 – 21.

See Also: The features described in this section are available only on some operating systems. This information is operating system–dependent; see your operating system–specific Oracle documentation.

Using Operating System Role Identification

To operate a database so that it uses the operating system to identify each user's database roles when a session is created, set the initialization parameter OS_ROLES to TRUE (and restart the instance, if it is currently running). When a user attempts to create a session with the database, Oracle initializes the user's security domain using the database roles identified by the operating system.

To identify database roles for a user, each Oracle user's operating system account must have operating system identifiers (these may be called groups, rights identifiers, or other similar names) that indicate which database roles are to be available for the user. Role specification can also indicate which roles are the default roles of a user and which roles are available with the ADMIN OPTION. No matter which operating system is used, the role specification at the operating system level follows the format:

```
ORA_<ID>_<ROLE>[_[D][A]]
```

where:

ID
The definition of ID varies on different operating systems. For example, on VMS, ID is the instance identifier of the database; on MVS, it is the machine type; on UNIX, it is the system ID.

D
This optional character indicates that this role is to be a default role of the database user.

A
This optional character indicates that this role is to be granted to the user with the ADMIN OPTION. This allows the user to grant the role to other roles only. (Roles cannot be granted to users if the operating system is used to manage roles.)

Note: If either the D or A characters are specified, they must be preceded by an underscore.

For example, an operating system account might have the following roles identified in its profile:

```
ORA_PAYROLL_ROLE1
ORA_PAYROLL_ROLE2_A
ORA_PAYROLL_ROLE3_D
ORA_PAYROLL_ROLE4_DA
```

When the corresponding user connects to the PAYROLL instance of Oracle, ROLE3 and ROLE4 are defaults, while ROLE2 and ROLE4 are available with the ADMIN OPTION.

Using Operating System Role Management

When you use operating system managed roles, it is important to note that database roles are being granted to an operating system user. Any database user to which the OS user is able to connect will have the authorized database roles enabled. For this reason, you should consider defining all Oracle users as IDENTIFIED EXTERNALLY if you are using OS_ROLES = TRUE, so that the database accounts are tied to the OS account that was granted privileges.

Granting and Revoking Roles When OS_ROLES=TRUE

If OS_ROLES is set to TRUE, the operating system completely manages the grants and revokes of roles *to users*. Any previous grants of roles to users via GRANT statements do not apply; however, they are still listed in the data dictionary. Only the role grants made at the operating system level to users apply. Users can still grant privileges to roles and users.

> **Note:** If the operating system grants a role to a user with the ADMIN OPTION, the user can grant the role only to other roles.

Enabling and Disabling Roles When OS_ROLES=TRUE

If OS_ROLES is set to TRUE, any role granted by the operating system can be dynamically enabled using the SET ROLE command. If the role was defined to require a password or operating system authorization, that still applies. However, any role not identified in a user's operating system account cannot be specified in a SET ROLE statement, even if a role has been granted using a GRANT statement when OS_ROLES = FALSE. (If you specify such a role, Oracle ignores it.)

When OS_ROLES = TRUE, a user can enable as many roles as specified by the parameter MAX_ENABLED_ROLES.

Using Network Connections with Operating System Role Management

If you want to have the operating system manage roles, by default users cannot connect to the database through the multi–threaded server. This restriction is the default because a remote user could impersonate another operating system user over a non–secure connection.

If you are not concerned with this security risk and want to use operating system role management with the multi–threaded server, or any other network connection, set the parameter REMOTE_OS_ROLES in the database's parameter file to TRUE. The change will take effect the next time you start the instance and mount the database. (The parameter is FALSE by default.)

Listing Privilege and Role Information

To list the grants made for objects, a user can query the following data dictionary views:

- ALL_COL_PRIVS, USER_COL_PRIVS, DBA_COL_PRIVS
- ALL_COL_PRIVS_MADE, USER_COL_PRIVS_MADE
- ALL_COL_PRIVS_RECD, USER_COL_PRIVS_RECD
- ALL_TAB_PRIVS, USER_TAB_PRIVS, DBA_TAB_PRIVS
- ALL_TAB_PRIVS_MADE, USER_TAB_PRIVS_MADE
- ALL_TAB_PRIVS_RECD, USER_TAB_PRIVS_RECD
- DBA_ROLES
- USER_ROLE_PRIVS, DBA_ROLE_PRIVS
- USER_SYS_PRIVS, DBA_SYS_PRIVS
- COLUMN_PRIVILEGES
- ROLE_ROLE_PRIVS, ROLE_SYS_PRIVS, ROLE_TAB_PRIVS
- SESSION_PRIVS, SESSION_ROLES

Note: See the *Oracle7 Server Reference* for a detailed description of these data dictionary views.

Listing Privilege and Role Information: Examples

For the following examples, assume the following statements are issued:

```
CREATE ROLE security_admin IDENTIFIED BY honcho;

GRANT create profile, alter profile, drop profile,
   create role, drop any role, grant any role, audit any,
   audit system, create user, become user, alter user, drop user
   TO security_admin WITH ADMIN OPTION;

GRANT SELECT, DELETE ON sys.aud$ TO security_admin;

GRANT security_admin, create session TO swilliams;

GRANT security_admin TO system_administrator;

GRANT create session TO jward;

GRANT SELECT, DELETE ON emp TO jward;

GRANT INSERT (ename, job) ON emp TO swilliams, jward;
```

Listing All System Privilege Grants

The following query indicates all system privilege grants made to roles and users:

```
SELECT * FROM sys.dba_sys_privs;

GRANTEE            PRIVILEGE                                ADM
------------------ ---------------------------------------  ---
SECURITY_ADMIN     ALTER PROFILE                            YES
SECURITY_ADMIN     ALTER USER                               YES
SECURITY_ADMIN     AUDIT ANY                                YES
SECURITY_ADMIN     AUDIT SYSTEM                             YES
SECURITY_ADMIN     BECOME USER                              YES
SECURITY_ADMIN     CREATE PROFILE                           YES
SECURITY_ADMIN     CREATE ROLE                              YES
SECURITY_ADMIN     CREATE USER                              YES
SECURITY_ADMIN     DROP ANY ROLE                            YES
SECURITY_ADMIN     DROP PROFILE                             YES
SECURITY_ADMIN     DROP USER                                YES
SECURITY_ADMIN     GRANT ANY ROLE                           YES
SWILLIAMS          CREATE SESSION                           NO
JWARD              CREATE SESSION                           NO
```

Listing All Role Grants

The following query returns all the roles granted to users and other roles:

```
SELECT * FROM sys.dba_role_privs;

GRANTEE            GRANTED_ROLE                                ADM
------------------ ------------------------------------------  ---
SWILLIAMS          SECURITY_ADMIN                              NO
```

Listing Object Privileges Granted to a User

The following query returns all object privileges (not including column specific privileges) granted to the specified user:

```
SELECT table_name, privilege, grantable FROM sys.dba_tab_privs
   WHERE grantee = 'JWARD';

TABLE_NAME   PRIVILEGE    GRANTABLE
------------ ------------ -----------
EMP          SELECT       NO
EMP          DELETE       NO
```

To list all the column specific privileges that have been granted, use the following query:

```
SELECT grantee, table_name, column_name, privilege
   FROM sys.dba_col_privs;

GRANTEE       TABLE_NAME    COLUMN_NAME         PRIVILEGE
------------- ------------- ------------------- -----------------
SWILLIAMS     EMP           ENAME               INSERT
SWILLIAMS     EMP           JOB                 INSERT
JWARD         EMP           ENAME               INSERT
JWARD         EMP           JOB                 INSERT
```

Listing the Current Privilege Domain of Your Session

The following query lists all roles currently enabled for the issuer:

```
SELECT * FROM session_roles;
```

If SWILLIAMS has enabled the SECURITY_ADMIN role and issues this query, Oracle returns the following information:

```
ROLE
------------------------------
SECURITY_ADMIN
```

The following query lists all system privileges currently available in the issuer's security domain, both from explicit privilege grants and from enabled roles:

```
SELECT * FROM session_privs;
```

If SWILLIAMS has the SECURITY_ADMIN role enabled and issues this query, Oracle returns the following results:

```
PRIVILEGE
----------------------------------------
AUDIT SYSTEM
CREATE SESSION
CREATE USER
BECOME USER
ALTER USER
DROP USER
CREATE ROLE
DROP ANY ROLE
GRANT ANY ROLE
AUDIT ANY
CREATE PROFILE
ALTER PROFILE
DROP PROFILE
```

If the SECURITY_ADMIN role is disabled for SWILLIAMS, the first query would have returned no rows, while the second query would only return a row for the CREATE SESSION privilege grant.

Listing Roles of the Database

The DBA_ROLES data dictionary view can be used to list all roles of a database and the authentication used for each role. For example, the following query lists all the roles in the database:

```
SELECT * FROM sys.dba_roles;

ROLE                           PASSWORD
------------------------------ --------
CONNECT                        NO
RESOURCE                       NO
DBA                            NO
SECURITY_ADMIN                 YES
```

Listing Information About the Privilege Domains of Roles

The ROLE_ROLE_PRIVS, ROLE_SYS_PRIVS, and ROLE_TAB_PRIVS data dictionary views contain information on the privilege domains of roles.

For example, the following query lists all the roles granted to the SYSTEM_ADMIN role:

```
SELECT granted_role, admin_option
   FROM role_role_privs
   WHERE role = ’SYSTEM_ADMIN’;
GRANTED_ROLE                   ADM
------------------------------ ---
SECURITY_ADMIN                 NO
```

The following query lists all the system privileges granted to the SECURITY_ADMIN role:

```
SELECT * FROM role_sys_privs WHERE role = ’SECURITY_ADMIN’;

ROLE                      PRIVILEGE                           ADM
------------------------- ----------------------------------- ---
SECURITY_ADMIN            ALTER PROFILE                       YES
SECURITY_ADMIN            ALTER USER                          YES
SECURITY_ADMIN            AUDIT ANY                           YES
SECURITY_ADMIN            AUDIT SYSTEM                        YES
SECURITY_ADMIN            BECOME USER                         YES
SECURITY_ADMIN            CREATE PROFILE                      YES
SECURITY_ADMIN            CREATE ROLE                         YES
SECURITY_ADMIN            CREATE USER                         YES
SECURITY_ADMIN            DROP ANY ROLE                       YES
SECURITY_ADMIN            DROP PROFILE                        YES
SECURITY_ADMIN            DROP USER                           YES
SECURITY_ADMIN            GRANT ANY ROLE                      YES
```

The following query lists all the object privileges granted to the SECURITY_ADMIN role:

```
SELECT table_name, privilege FROM role_tab_privs
   WHERE role = ’SECURITY_ADMIN’;

TABLE_NAME                     PRIVILEGE
------------------------------ -------------------
AUD$                           DELETE
AUD$                           SELECT
```

CHAPTER

21

Auditing Database Use

This chapter describes how to use the Oracle auditing facilities, and includes the following topics:

- Guidelines for Auditing
- Creating and Deleting the Database Audit Trail Views
- Managing Audit Trail Information
- Viewing Database Audit Trail Information
- Auditing through Database Triggers

See Also: If you are using Trusted Oracle7, see the *Trusted Oracle7 Server Administrator's Guide* for additional information about auditing and audit trail management.

Guidelines for Auditing

This section describes guidelines for auditing and includes the following topics:

- Audit via the Database or Operating System
- Keep Audited Information Manageable

Audit via the Database or Operating System

The data dictionary of every database has a table named SYS.AUD$, commonly referred to as the database *audit trail*.

Either the database or operating system audit trail can store all audit records generated as the result of statement, privilege, or object auditing.

Your operating system may or may not support database auditing to the operating system audit trail. If this option is available, consider the advantages and disadvantages of using either the database or operating system auditing trail to store database audit records.

Using the database audit trail offers the following advantages:

- You can view selected portions of the audit trail with the predefined audit trail views of the data dictionary.
- You can use Oracle tools (such as ReportWriter) to generate audit reports.

Alternatively, your operating system audit trail may allow you to consolidate audit records from multiple sources including Oracle and other applications. Therefore, examining system activity might be more efficient because all audit records are in one place.

See Also: Your operating system may also contain an audit trail that stores audit records generated by the operating system auditing facility. However, this facility is operating system–dependent. See your operating system–specific Oracle documentation.

Keep Audited Information Manageable

Although auditing is relatively inexpensive, limit the number of audited events as much as possible. This will minimize the performance impact on the execution of statements that are audited, and minimize the size of the audit trail.

Use the following general guidelines when devising an auditing strategy:

- Evaluate your purpose for auditing.

  After you have a clear understanding of the reasons for auditing, you can devise an appropriate auditing strategy and avoid unnecessary auditing.

  For example, suppose you are auditing to investigate suspicious database activity. This information by itself is not specific enough. What types of suspicious database activity do you suspect or have you noticed? A more focused auditing purpose might be to audit unauthorized deletions from arbitrary tables in the database. This purpose narrows the type of action being audited and the type of object being affected by the suspicious activity.

- Audit knowledgeably.

  Audit the minimum number of statements, users, or objects required to get the targeted information. This prevents unnecessary audit information from cluttering the meaningful information and consuming valuable space in the SYSTEM tablespace. Balance your need to gather sufficient security information with your ability to store and process it.

  For example, if you are auditing to gather information about database activity, determine exactly what types of activities you are tracking, audit only the activities of interest, and audit only for the amount of time necessary to gather the information you desire. Do not audit objects if you are only interested in each session's logical I/O information.

Auditing Suspicious Database Activity

When you audit to monitor suspicious database activity, use the following guidelines:

- Audit generally, then specifically.

  When starting to audit for suspicious database activity, it is common that not much information is available to target specific users or schema objects. Therefore, audit options must be set more generally at first. Once preliminary audit information is recorded and analyzed, the general audit options should be turned off and more specific audit options enabled. This process should continue until enough evidence is gathered to make concrete conclusions about the origin of the suspicious database activity.

- Protect the audit trail.

When auditing for suspicious database activity, protect the audit trail so that audit information cannot be added, changed, or deleted without being audited.

See Also: For more information about the audit trail, see "Protecting the Audit Trail" on page 21 – 18.

Auditing Normal Database Activity

When your purpose for auditing is to gather historical information about particular database activities, use the following guidelines:

- Audit only pertinent actions.

  To avoid cluttering meaningful information with useless audit records and reduce the amount of audit trail administration, only audit the targeted database activities.

- Archive audit records and purge the audit trail.

  After you have collected the required information, archive the audit records of interest and purge the audit trail of this information.

Creating and Deleting the Database Audit Trail Views

This section describes how to create and delete database audit trail views, and includes the following topics:

- Creating the Audit Trail Views
- Deleting the Audit Trail Views

The database audit trail (SYS.AUD$) is a single table in each Oracle database's data dictionary. To help you view meaningful auditing information in this table, several predefined views are provided. They must be created for you to use auditing; you can later delete them if you decide not to use auditing.

See Also: On most operating systems, the audit trail views are created automatically with the data dictionary. See your operating system–specific Oracle documentation.

Creating the Audit Trail Views

If you decide to use auditing, create the auditing views by connecting as SYS and running the script CATAUDIT.SQL. This script creates the following views:

- STMT_AUDIT_OPTION_MAP
- AUDIT_ACTIONS
- ALL_DEF_AUDIT_OPTS
- DBA_STMT_AUDIT_OPTS
- USER_OBJ_AUDIT_OPTS, DBA_OBJ_AUDIT_OPTS
- USER_AUDIT_TRAIL, DBA_AUDIT_TRAIL
- USER_AUDIT_SESSION, DBA_AUDIT_SESSION
- USER_AUDIT_STATEMENT, DBA_AUDIT_STATEMENT
- USER_AUDIT_OBJECT, DBA_AUDIT_OBJECT
- DBA_AUDIT_EXISTS
- USER_AUDIT_SESSION, DBA_AUDIT_SESSION
- USER_TAB_AUDIT_OPTS

See Also: For information about these views, see the *Oracle7 Server Reference.*

For examples of audit information interpretations, see "Viewing Database Audit Trail Information" on page 21 – 18.

Deleting the Audit Trail Views

If you disable auditing and no longer need the audit trail views, delete them by connecting to the database as SYS and running the script file CATNOAUD.SQL. The name and location of the CATNOAUD.SQL script are operating system–dependent.

Managing Audit Trail Information

This section describes various aspects of managing audit trail information, and includes the following topics:

- Events Audited by Default
- Setting Audit Options
- Enabling and Disabling Database Auditing
- Controlling the Growth and Size of the Audit Trail
- Protecting the Audit Trail

Depending on the events audited and the auditing options set, the audit trail records can contain different types of information. The following information is always included in each audit trail record, provided that the information is meaningful to the particular audit action:

- user name
- session identifier
- terminal identifier
- name of the object accessed
- operation performed or attempted
- completion code of the operation
- date and time stamp
- system privileges (including MAC privileges for Trusted Oracle7) used
- label of the user session (for Trusted Oracle7 only)
- label of the object accessed (for Trusted Oracle7 only)

Audit trail records written to the operating system audit trail contain some encodings that are not readable. These can be decoded as follows:

Action Code
This describes the operation performed or attempted. The AUDIT_ACTIONS data dictionary table contains a list of these codes and their descriptions.

Privileges Used
This describes any system privileges used to perform the operation. The SYSTEM_PRIVILEGE_MAP table lists all of these codes, and their descriptions.

Completion Code
This describes the result of the attempted operation. Successful operations return a value of zero, while unsuccessful operations return the Oracle error code describing why the operation was unsuccessful.

See Also: Error codes are listed in the *Oracle7 Server Messages* manual.

Events Audited by Default

Regardless of whether database auditing is enabled, the Oracle Server will always audit certain database–related actions into the operating system audit trail. These events include the following:

instance startup
An audit record is generated that details the OS user starting the instance, his terminal identifier, the

date and time stamp, and whether database auditing was enabled or disabled. This is audited into the OS audit trail because the database audit trail is not available until after startup has successfully completed. Recording the state of database auditing at startup further prevents an administrator from restarting a database with database auditing disabled so that they are able to perform unaudited actions.

instance shutdown
An audit record is generated that details the OS user shutting down the instance, her terminal identifier, the date and time stamp.

connections to the database with administrator privileges
An audit record is generated that details the OS user connecting to Oracle as SYSOPER or SYSDBA, to provide accountability of users with administrator privileges.

On operating systems that do not make an audit trail accessible to Oracle, these audit trail records are placed in an Oracle audit trail file in the same directory as background process trace files.

Setting Auditing Options

Depending on the auditing options set, audit records can contain different types of information. However, all auditing options generate the following information:

- the user that executed the audited statement
- the action code (a number) that indicates the audited statement executed by the user
- the object or objects referenced in the audited statement
- the date and time that the audited statement was executed

The audit trail does not store information about any data values that might be involved in the audited statement. For example, old and new data values of updated rows are not stored when an UPDATE statement is audited. However, this specialized type of auditing can be performed on DML statements involving tables by using database triggers.

Oracle allows you to set audit options at three levels:

statement — audits based on the type of a SQL statement, such as any SQL statement on a table (which records each CREATE, TRUNCATE, and DROP TABLE statement)

privilege — audits use of a particular system privilege, such as CREATE TABLE

object — audits specific statements on specific objects, such as ALTER TABLE on the EMP table

See Also: For examples of trigger usage for this specialized type of auditing, see page 21 – 21.

Statement Audit Options

Valid statement audit options that can be included in AUDIT and NOAUDIT statements are listed in Table 21 – 1.

Option	SQL Statements Audited
ALTER SYSTEM	ALTER SYSTEM
CLUSTER	CREATE CLUSTER ALTER CLUSTER TRUNCATE CLUSTER DROP CLUSTER
DATABASE LINK	CREATE DATABASE LINK DROP DATABASE LINK
INDEX	CREATE INDEX ALTER INDEX DROP INDEX
NOT EXISTS	All SQL statements that return an Oracle error because the specified structure or object does not exist

PROCEDURE	CREATE [OR REPLACE] FUNCTION
	CREATE [OR REPLACE] PACKAGE
	CREATE [OR REPLACE] PACKAGE BODY
	CREATE [OR REPLACE] PROCEDURE
	DROP PACKAGE
	DROP PROCEDURE

PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM

ROLE	CREATE ROLE
	ALTER ROLE
	SET ROLE
	DROP ROLE

ROLLBACK SEGMENT	CREATE ROLLBACK SEGMENT ALTER DROPBACK SEGMENT‘DROP ROLLBACK SEGMENT

Table 21 – 1 Statement Auditing Options, continued on next page

Option	SQL Statements Audited
SEQUENCE	CREATE SEQUENCE DROP SEQUENCE
SESSION	Connects and Disconnects
SYNONYM	CREATE SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT NO AUDIT
SYSTEM GRANT	GRANT system privileges/role TO user/role REVOKE system privileges/role FROM user/role
TABLE	CREATE TABLE ALTER TABLE DROP TABLE
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE
TRIGGER	CREATE TRIGGER ALTER TRIGGER ENABLE or DISABLE ALTER TABLE with ENABLE, DISABLE, and DROP clauses
USER	CREATE USER ALTER USER DROP USER
VIEW	CREATE [OR REPLACE] VIEW DROP VIEW

Table 21 – 1 Statement Auditing Options

Shortcuts for Statement Audit Options Shortcuts are provided so that you can specify several related statement options with one word.

Shortcuts are not statement options themselves; rather, they are ways of specifying sets of related statement options with one word in AUDIT and NOAUDIT statements.

CONNECT	equivalent to the SESSION option
RESOURCE	equivalent to the options ALTER SYSTEM, CLUSTER, DATABASE LINK, PROCEDURE, ROLLBACK SEGMENT, SEQUENCE, SYNONYM, TABLE, TABLESPACE, and VIEW
DBA	equivalent to the options SYSTEM AUDIT, PUBLIC DATABASE LINK, PUBLIC SYNONYM, ROLE, SYSTEM GRANT, and USER
ALL	equivalent to all options in Table 21 – 1, including the NOT EXISTS option

Warning: Do not confuse the shortcuts CONNECT, RESOURCE, and DBA with the predefined roles of the same names.

Auditing Connections and Disconnections

The SESSION statement option (and CONNECT shortcut) is unique because it does not generate an audit record when a particular type of statement is issued; this option generates a single audit record for each session created by connections to an instance. An audit record is inserted into the audit trail at connect time and updated at disconnect time. Cumulative information about a session such as connection time, disconnection time, logical and physical I/Os processed, and more is stored in a single audit record that corresponds to the session.

Table 21 – 2 lists additional audit options not covered by any of the above shortcuts.

Object Option	SQL Statements Audited
ALTER SEQUENCE	ALTER SEQUENCE sequence
ALTER TABLE	ALTER TABLE table
COMMENT TABLE	COMMENT ON table, view, snapshot, column
DELETE TABLE	DELETE FROM table, view
EXECUTE PROCEDURE	Calls to procedures and functions
GRANT PROCEDURE	GRANT privilege ON procedure REVOKE privilege ON sequence
GRANT TABLE	GRANT privilege ON table, view, snapshot REVOKE privilege ON table, view, snapshot
INSERT TABLE	INSERT INTO table view
LOCK TABLE	LOCK TABLE table, view
SELECT SEQUENCE	Reference to a sequence
SELECT TABLE	SELECT . . .FROM table, view, snapshot
UPDATE TABLE	UPDATE table, view

Table 21 – 2 Statement Auditing Options

Privilege Audit Options

Privilege audit options exactly match the corresponding system privileges. For example, the option to audit use of the DELETE ANY TABLE privilege is DELETE ANY TABLE. To turn this option on, you would use a statement similar to the following example:

```
AUDIT DELETE ANY TABLE
   BY ACCESS
   WHENEVER NOT SUCCESSFUL;
```

Oracle’s system privileges are listed beginning on page 20 – 2.

Object Audit Options

Table 21 – 3 lists valid object audit options and the schema object types for which each option is available.

Object Option	Table	View	Sequence	Procedure[1]
ALTER	✓		✓	
AUDIT	✓	✓	✓	✓
COMMENT	✓	✓		
DELETE	✓	✓		
EXECUTE				✓
GRANT	✓	✓	✓	✓
INDEX	✓			
INSERT	✓	✓		
LOCK	✓	✓		
RENAME	✓	✓		✓
SELECT	✓	✓[2]	✓	
UPDATE	✓	✓		

Table 21 – 3 Object Audit Options

[1] *"Procedure" refers to stand–alone stored procedures and functions, and packages.*
[2] *The SELECT option may also be used for snapshots.*

Table 21 – 4 lists the SQL statements audited by each object option.

Object Option	Table
ALTER	ALTER object (table or sequence)
AUDIT	AUDIT (Form II) object
COMMENT	COMMENT object (table or view)
DELETE	DELETE FROM object (table or view)
EXECUTE	EXECUTE object (procedure[1])
GRANT	GRANT (Form II) privilege ON object
INDEX	CREATE INDEX ON object (tables only)
INSERT	INSERT INTO object (table, view, or procedure)
LOCK	LOCK object (table or view)
RENAME	RENAME object (table, view, or procedure[1])
SELECT	SELECT . . .FROM object (table, view, snapshot)
UPDATE	UPDATE object (table or view)

Table 21 – 4 SQL Statement Audited by Database Object Audit Options

[1] *Procedure refers to stand–alone stored procedures and functions, and packages.*

Shortcut for Object Audit Options The ALL shortcut can be used to specify all available object audit options for a schema object. This shortcut is not an option itself; rather, it is a way of specifying all object audit options with one word in AUDIT and NOAUDIT statements.

Enabling Audit Options

The SQL command AUDIT turns on statement and privilege audit options, and object audit options. Audit statements that set statement and privilege audit options can include the BY USER option to specify a list of users to limit the scope of the statement and privilege audit options. The SQL command AUDIT turns on audit options. To use it to set statement and privilege options, you must have the AUDIT SYSTEM privilege. To use it to set object audit options, you must own the object to be audited or have the AUDIT ANY privilege.

You can set any auditing option, and specify the following conditions for auditing:

- WHENEVER SUCCESSFUL / WHENEVER NOT SUCCESSFUL
- BY SESSION / BY ACCESS

A new database session picks up auditing options from the data dictionary when the session is created. These auditing options remain in force for the duration of the database connection. Setting new system or object auditing options causes all subsequent database sessions to use these options; existing sessions will continue using the audit options in place at session creation.

⚠ **Warning:** The AUDIT command only turns auditing options on; it does not enable auditing as a whole. To turn auditing on and control whether Oracle generates audit records based on the audit options currently set, set the parameter AUDIT_TRAIL in the database's parameter file.

The following examples illustrate the use of the AUDIT command.

See Also: For a complete description of the AUDIT command, see the *Oracle7 Server SQL Reference.*

For more information about enabling and disabling auditing, see "Enabling and Disabling Database Auditing" on page 21 – 15.

Enabling Statement Privilege Auditing To audit all successful and unsuccessful connections to and disconnections from the database, regardless of user, BY SESSION (the default and only value for this option), enter the following statement:

```
AUDIT SESSION;
```

You can set this option selectively for individual users also, as in the next example:

```
AUDIT SESSION
   BY scott, lori;
```

To audit all successful and unsuccessful uses of the DELETE ANY TABLE system privilege, enter the following statement:

```
AUDIT DELETE ANY TABLE;
```

To audit all unsuccessful SELECT, INSERT, and DELETE statements on all tables and unsuccessful uses of the EXECUTE ANY PROCEDURE system privilege, by all database users, BY ACCESS, enter the following statement:

```
AUDIT SELECT TABLE, INSERT TABLE, DELETE TABLE,
   EXECUTE ANY PROCEDURE
      BY ACCESS
      WHENEVER NOT SUCCESSFUL;
```

The AUDIT SYSTEM system privilege is required to set any statement or privilege audit option. Normally, the security administrator is the only user granted this system privilege.

Enabling Object Auditing To audit all successful and unsuccessful DELETE statements on the EMP table, BY SESSION (the default value), enter the following statement:

```
AUDIT DELETE ON emp;
```

To audit all successful SELECT, INSERT, and DELETE statements on the DEPT table owned by user JWARD, BY ACCESS, enter the following statement:

```
AUDIT SELECT, INSERT, DELETE
   ON jward.dept
   BY ACCESS
   WHENEVER SUCCESSFUL;
```

To set the default object auditing options to audit all unsuccessful SELECT statements, BY SESSION (the default), enter the following statement:

```
AUDIT SELECT
   ON DEFAULT
   WHENEVER NOT SUCCESSFUL;
```

A user can set any object audit option for the objects contained in the user’s schema. The AUDIT ANY system privilege is required to set an object audit option for an object contained in another user’s schema or to set the default object auditing options; normally, the security administrator is the only user granted this system privilege.

Disabling Audit Options

The NOAUDIT command turns off the various audit options of Oracle. Use it to reset statement and privilege audit options, and object audit options. A NOAUDIT statement that sets statement and privilege audit options can include the BY USER option to specify a list of users to limit the scope of the statement and privilege audit options.

You can use a NOAUDIT statement to disable an audit option selectively using the WHENEVER clause. If the clause is not specified, the auditing option is disabled entirely, for both successful and non–successful cases.

The BY SESSION / BY ACCESS option pair is *not* supported by the NOAUDIT command; audit options, no matter how they were turned on, are turned off by an appropriate NOAUDIT statement.

The following examples illustrate the use of the NOAUDIT command.

> **Warning:** The NOAUDIT command only turns auditing options off; it does not disable auditing as a whole. To turn auditing off and stop Oracle from generating audit records, even though you have audit options currently set, set the parameter AUDIT_TRAIL in the database's parameter file.

See Also: For a complete syntax listing of the NOAUDIT command, see the *Oracle7 Server SQL Reference.*

Also see "Enabling and Disabling Database Auditing" on page 21 – 15.

Disabling Statement and Privilege Auditing The following statements turn off the corresponding audit options:

```
NOAUDIT session;
NOAUDIT session BY scott, lori;
NOAUDIT DELETE ANY TABLE;
NOAUDIT SELECT TABLE, INSERT TABLE, DELETE TABLE,
   EXECUTE ANY PROCEDURE;
```

The following statements turn off all statement (system) and privilege audit options:

```
NOAUDIT ALL;
NOAUDIT ALL PRIVILEGES;
```

To disable statement or privilege auditing options, you must have the AUDIT SYSTEM system privilege.

Disabling Object Auditing The following statements turn off the corresponding auditing options:

```
NOAUDIT DELETE
   ON emp;
NOAUDIT SELECT, INSERT, DELETE
   ON jward.dept;
```

Furthermore, to turn off all object audit options on the EMP table, enter the following statement:

```
NOAUDIT ALL
   ON emp;
```

Disabling Default Object Audit Options To turn off all default object audit options, enter the following statement:

```
NOAUDIT ALL
   ON DEFAULT;
```

Note that all schema objects created before this NOAUDIT statement is issued continue to use the default object audit options in effect at the time of their creation, unless overridden by an explicit NOAUDIT statement after their creation.

To disable object audit options for a specific object, you must be the owner of the schema object. To disable the object audit options of an object in another user's schema or to disable default object audit options, you must have the AUDIT ANY system privilege. A user with privileges to disable object audit options of an object can override the options set by any user.

Enabling and Disabling Database Auditing

Any authorized database user can set statement, privilege, and object auditing options at any time, but Oracle does not generate and store audit records in the audit trail unless database auditing is enabled. The security administrator is normally responsible for this operation.

Database auditing is enabled and disabled by the AUDIT_TRAIL initialization parameter in the database's parameter file. The parameter can be set to the following values:

DB	enables database auditing and directs all audit records to the database audit trail
OS	enables database auditing and directs all audit records to the operating system audit trail
NONE	disables auditing (This value is the default.)

Once you have edited the parameter file, restart the database instance to enable or disable database auditing as intended.

See Also: For more information about editing parameter files, see the *Oracle7 Server Reference.*

Controlling the Growth and Size of the Audit Trail

If the audit trail becomes completely full and no more audit records can be inserted, audited statements cannot be successfully executed until the audit trail is purged. Warnings are returned to all users that issue audited statements. Therefore, the security administrator must control the growth and size of the audit trail.

When auditing is enabled and audit records are being generated, the audit trail grows according to two factors:

- the number of audit options turned on
- the frequency of execution of audited statements

To control the growth of the audit trail, you can use the following methods:

- Enable and disable database auditing. If it is enabled, audit records are generated and stored in the audit trail; if it is disabled, audit records are not generated.
- Be very selective about the audit options that are turned on. If more selective auditing is performed, useless or unnecessary audit information is not generated and stored in the audit trail.
- Tightly control the ability to perform object auditing. This can be done two different ways:
  - A security administrator owns all objects and the AUDIT ANY system privilege is never granted to any other user. Alternatively, all schema objects can belong to a schema for which the corresponding user does not have CREATE SESSION privilege.
  - All objects are contained in schemas that do not correspond to real database users (that is, the CREATE SESSION privilege is not granted to the corresponding user) and the security administrator is the only user granted the AUDIT ANY system privilege.

  In both scenarios, object auditing is controlled entirely by the security administrator.

The maximum size of the database audit trail (SYS.AUD$ table) is predetermined during database creation. By default, up to 99 extents, each 10K in size, can be allocated for this table.

See Also: If you are directing audit records to the operating system audit trail, see your operating system–specific Oracle documentation for more information about managing the operating system audit trail.

Purging Audit Records from the Audit Trail

After auditing is enabled for some time, the security administrator may want to delete records from the database audit trail both to free audit trail space and to facilitate audit trail management.

For example, to delete *all* audit records from the audit trail, enter the following statement:

```
DELETE FROM sys.aud$;
```

Alternatively, to delete all audit records from the audit trail generated as a result of auditing the table EMP, enter the following statement:

```
DELETE FROM sys.aud$
   WHERE obj$name='EMP';
```

If audit trail information must be archived for historical purposes, the security administrator can copy the relevant records to a normal database table (for example, using "INSERT INTO table SELECT ... FROM sys.aud$ ...") or export the audit trail table to an operating system file.

Only the user SYS, a user who has the DELETE ANY TABLE privilege, or a user to whom SYS has granted DELETE privilege on SYS.AUD$ can delete records from the database audit trail.

> **Note:** If the audit trail is completely full and connections are being audited (that is, if the SESSION option is set), typical users cannot connect to the database because the associated audit record for the connection cannot be inserted into the audit trail. In this case, the security administrator must connect as SYS (operations by SYS are not audited) and make space available in the audit trail.

See Also: For information about exporting tables, see the *Oracle7 Server Utilities* guide.

Reducing the Size of the Audit Trail

As with any database table, after records are deleted from the database audit trail, the extents allocated for this table still exist.

If the database audit trail has many extents allocated for it, but many of them are not being used, the space allocated to the database audit trail can be reduced using the following steps:

To Reduce the Size of the Audit Trail

1. If you want to save information currently in the audit trail, copy it to another database table or export it using the EXPORT utility.
2. Connect as with administrator privileges.
3. Truncate SYS.AUD$ using the TRUNCATE command.

4. Reload archived audit trail records generated from Step 1.

The new version of SYS.AUD$ is allocated only as many extents that are necessary to contain current audit trail records.

> **Note:** SYS.AUD$ is the only SYS object that should ever be directly modified.

Protecting the Audit Trail

When auditing for suspicious database activity, protect the integrity of the audit trail's records to guarantee the accuracy and completeness of the auditing information.

To protect the database audit trail from unauthorized deletions, grant the DELETE ANY TABLE system privilege to security administrators only.

To audit changes made to the database audit trail, use the following statement:

```
AUDIT INSERT, UPDATE, DELETE
   ON sys.aud$
   BY ACCESS;
```

Audit records generated as a result of object audit options set for the SYS.AUD$ table can only be deleted from the audit trail by someone connected with administrator privileges, which itself has protection against unauthorized use. As a final measure of protecting the audit trail, any operation performed while connected with administrator privileges is audited in the operating system audit trail, if available.

See Also: For more information about the availability of an operating system audit trail and possible uses, see your operating system–specific Oracle documentation.

Viewing Database Audit Trail Information

This section offers examples that demonstrate how to examine and interpret the information in the audit trail, and includes the following topics:

- Listing Active Statement Audit Options
- Listing Active Privilege Audit Options
- Listing Active Object Audit Options for Specific Objects

- Listing Default Object Audit Options
- Listing Audit Records
- Listing Audit Records for the AUDIT SESSION Option

You may have to audit a database for the following suspicious activities:

- Passwords, tablespace settings, and quotas for some database users are being altered without authorization.
- A high number of deadlocks are occurring, most likely because of users acquiring exclusive table locks.
- Rows are arbitrarily being deleted from the EMP table in SCOTT's schema.

As an example, say that you suspect the users JWARD and SWILLIAMS of several of these detrimental actions. The database administrator may then issue the following statements (in order):

```
AUDIT ALTER, INDEX, RENAME ON DEFAULT
   BY SESSION;
CREATE TABLE scott.emp . . . ;
CREATE VIEW scott.employee AS SELECT * FROM scott.emp;
AUDIT SESSION BY jward, swilliams;
AUDIT ALTER USER;
AUDIT LOCK TABLE
   BY ACCESS
   WHENEVER SUCCESSFUL;
AUDIT DELETE ON scott.emp
   BY ACCESS
   WHENEVER SUCCESSFUL;
```

The following statements are subsequently issued by the user JWARD:

```
ALTER USER tsmith QUOTA 0 ON users;
DROP USER djones;
```

The following statements are subsequently issued by the user SWILLIAMS:

```
LOCK TABLE scott.emp IN EXCLUSIVE MODE;
DELETE FROM scott.emp WHERE mgr = 7698;
ALTER TABLE scott.emp ALLOCATE EXTENT (SIZE 100K);
CREATE INDEX scott.ename_index ON scott.emp (ename);
CREATE PROCEDURE scott.fire_employee (empid NUMBER) AS
BEGIN
   DELETE FROM scott.emp WHERE empno = empid;
END;
/
EXECUTE scott.fire_employee(7902);
```

The following sections show the information that can be listed using the audit trail views in the data dictionary.

Listing Active Statement Audit Options

The following query returns all the statement audit options that are set:

```
SELECT * FROM sys.dba_stmt_audit_opts;

USER_NAME            AUDIT_OPTION         SUCCESS    FAILURE
-------------------- -------------------- ---------- ----------
JWARD                SESSION              BY SESSION BY SESSION
SWILLIAMS            SESSION              BY SESSION BY SESSION
                     LOCK TABLE           BY ACCESS  NOT SET
```

Notice that the view reveals the statement audit options set, whether they are set for success or failure (or both), and whether they are set for BY SESSION or BY ACCESS.

Listing Active Privilege Audit Options

The following query returns all the privilege audit options that are set:

```
SELECT * FROM sys.dba_priv_audit_opts;

USER_NAME            AUDIT_OPTION         SUCCESS    FAILURE
-------------------- -------------------- ---------- ----------
ALTER USER           BY SESSION BY SESSION
```

Listing Active Object Audit Options for Specific Objects

The following query returns all audit options set for any objects contained in SCOTT’s schema:

```
SELECT * FROM sys.dba_obj_audit_opts
   WHERE owner = ’SCOTT’ AND object_name LIKE ’EMP%’;

OWNER  OBJECT_NAME OBJECT_TY ALT AUD COM DEL GRA IND INS LOC ...
------ ----------- --------- --- --- --- --- --- --- --- --- ...
SCOTT  EMP         TABLE     S/S -/- -/- A/- -/- S/S -/- -/- ...
SCOTT  EMPLOYEE    VIEW      -/- -/- -/- -/- -/- -/- -/- -/- ...
```

Notice that the view returns information about all the audit options for the specified object. The information in the view is interpreted as follows:

- The character “–” indicates that the audit option is not set.
- The character “S” indicates that the audit option is set, BY SESSION.
- The character “A” indicates that the audit option is set, BY ACCESS.
- Each audit option has two possible settings, WHENEVER SUCCESSFUL and WHENEVER NOT SUCCESSFUL, separated by “/”. For example, the DELETE audit option for SCOTT.EMP is

set BY ACCESS for successful delete statements and not set at all for unsuccessful delete statements.

Listing Default Object Audit Options

The following query returns all default object audit options:

```
SELECT * FROM all_def_audit_opts;

ALT AUD COM DEL GRA IND INS LOC REN SEL UPD REF EXE
--- --- --- --- --- --- --- --- --- --- --- --- ---
S/S -/- -/- -/- -/- S/S -/- -/- S/S -/- -/- -/- -/-
```

Notice that the view returns information similar to the USER_OBJ_AUDIT_OPTS and DBA_OBJ_AUDIT_OPTS views (see previous example).

Listing Audit Records

The following query lists audit records generated by statement and object audit options:

```
SELECT username, obj_name, action_name, ses_actions
   FROM sys.dba_audit_object;
```

Listing Audit Records for the AUDIT SESSION Option

The following query lists audit information corresponding to the AUDIT SESSION statement audit option:

```
SELECT username, logoff_time, logoff_lread, logoff_pread,
   logoff_lwrite, logoff_dlock
   FROM sys.dba_audit_session;

USERNAME   LOGOFF_TI LOGOFF_LRE LOGOFF_PRE LOGOFF_LWR LOGOFF_DLO
---------- --------- ---------- ---------- ---------- ----------
JWARD      02-AUG-91         53          2         24 0
SWILLIAMS  02-AUG-91       3337        256        630 0
```

Auditing Through Database Triggers

You can use triggers to supplement the built–in auditing features of Oracle. Although you can write triggers to record information similar to that recorded by the AUDIT command, do so only when you need more detailed audit information. For example, you can use triggers to provide value–based auditing on a per–row basis for tables.

> **Note:** In some fields, the Oracle AUDIT command is considered a *security* audit facility, while triggers can provide a *financial* audit facility.

When deciding whether to create a trigger to audit database activity, consider the advantages that the standard Oracle database auditing features provide compared to auditing by triggers:

- Standard auditing options cover DML and DDL statements regarding all types of schema objects and structures. In contrast, triggers can audit only DML statements issued against tables.
- All database audit information is recorded centrally and automatically using the auditing features of Oracle.
- Auditing features enabled using the standard Oracle features are easier to declare and maintain and less prone to errors than are auditing functions defined through triggers.
- Any changes to existing auditing options can also be audited to guard against malicious database activity.
- Using the database auditing features, you can generate records once every time an audited statement is issued (BY ACCESS) or once for every session that issues an audited statement (BY SESSION). Triggers cannot audit by session; an audit record is generated each time a trigger–audited table is referenced.
- Database auditing can audit unsuccessful data access. In comparison, any audit information generated by a trigger is rolled back if the triggering statement is rolled back.
- Connections and disconnections, as well as session activity (such as physical I/Os, logical I/Os, and deadlocks), can be recorded by standard database auditing.

When using triggers to provide sophisticated auditing, normally use AFTER triggers. By using AFTER triggers, you record auditing information after the triggering statement is subjected to any applicable integrity constraints, preventing cases where audit processing is carried out unnecessarily for statements that generate exceptions to integrity constraints.

When you should use AFTER row vs. AFTER statement triggers depends on the information being audited. For example, row triggers provide value–based auditing on a per–row basis for tables. Triggers can also allow the user to supply a “reason code” for issuing the audited SQL statement, which can be useful in both row and statement–level auditing situations.

The following trigger audits modifications to the EMP table on a per–row basis. It requires that a “reason code” be stored in a global package variable before the update. The trigger demonstrates the following:

- how triggers can provide value–based auditing
- how to use public package variables

Comments within the code explain the functionality of the trigger.

```
CREATE TRIGGER audit_employee
AFTER INSERT OR DELETE OR UPDATE ON emp
FOR EACH ROW
BEGIN
/* AUDITPACKAGE is a package with a public package
   variable REASON. REASON could be set by the
   application by a command such as EXECUTE
   AUDITPACKAGE.SET_REASON(reason_string). Note that a
   package variable has state for the duration of a
   session and that each session has a separate copy of
   all package variables. */
IF auditpackage.reason IS NULL THEN
   raise_application_error(–20201,’Must specify reason with ’,
   ’AUDITPACKAGE.SET_REASON(reason_string)’);
END IF;

/* If the above conditional evaluates to TRUE, the
   user–specified error number and message is raised,
   the trigger stops execution, and the effects of the
   triggering statement are rolled back. Otherwise, a
   new row is inserted into the pre–defined auditing
   table named AUDIT_EMPLOYEE containing the existing
   and new values of the EMP table and the reason code
   defined by the REASON variable of AUDITPACKAGE. Note
   that the ”old” values are NULL if triggering
   statement is an INSERT and the ”new” values are NULL
   if the triggering statement is a DELETE. */
INSERT INTO audit_employee VALUES
   (:old.ssn, :old.name, :old.job_classification, :old.sal,
   :new.ssn, :new.name, :new.job_classification, :new.sal,
   auditpackage.reason, user, sysdate );
END;
```

Optionally, you can also set the reason code back to NULL if you want to force the reason code to be set for every update. The following AFTER statement trigger sets the reason code back to NULL after the triggering statement is executed:

```
CREATE TRIGGER audit_employee_reset
AFTER INSERT OR DELETE OR UPDATE ON emp
BEGIN
   auditpackage.set_reason(NULL);
END;
```

The previous two triggers are both fired by the same type of SQL statement. However, the AFTER row trigger is fired once for each row of the table affected by the triggering statement, while the AFTER statement trigger is fired only once after the triggering statement execution is completed.

PART

V

Database Backup and Recovery

CHAPTER

22

Archiving Redo Information

This chapter describes how to create and maintain the archived redo log, and includes the following topics:

- Choosing Between NOARCHIVELOG and ARCHIVELOG Mode
- Turning Archiving On and Off
- Tuning Archiving
- Displaying Archiving Status Information
- Specifying the Archived Redo Log Filename Format and Destination

See Also: If you are using Oracle with the Parallel Server, see the *Oracle7 Parallel Server Concepts & Administration* for additional information about archiving in the environment.

This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide.*

Choosing Between NOARCHIVELOG and ARCHIVELOG Mode

This section describes the issues you must consider when choosing to run your database in NOARCHIVELOG or ARCHIVELOG mode, and includes the following topics:

- Running a Database in NOARCHIVELOG Mode
- Running a Database in ARCHIVELOG Mode

Running a Database in NOARCHIVELOG Mode

When you run your database in NOARCHIVELOG mode, the archiving of the online redo log is disabled. Information in the database's control file indicates that filled groups are not required to be archived. Therefore, after a filled group becomes inactive and the checkpoint at the log switch completes, the group is available for reuse by LGWR.

NOARCHIVELOG mode protects a database only from instance failure, not from disk (media) failure. Only the most recent changes made to the database, stored in the groups of the online redo log, are available for instance recovery. In other words, if you are using NOARCHIVELOG mode, you can only *restore* (not recover) the database to the point of the most recent full database backup. You cannot recover subsequent transactions.

Also, in NOARCHIVELOG mode, you cannot perform online tablespace backups. Furthermore, you cannot use online tablespace backups previously taken while the database operated in ARCHIVELOG mode. Only full backups taken while the database is closed can be used to restore a database operating in NOARCHIVELOG mode. Therefore, if you decide to operate a database in NOARCHIVELOG mode, take full database backups at regular, frequent intervals.

Running a Database in ARCHIVELOG Mode

When you run a database in ARCHIVELOG mode, the archiving of the online redo log is enabled. Information in a database control file indicates that a group of filled online redo log files cannot be used by LGWR until the group is archived. A filled group is immediately available to the process performing the archiving after a log switch occurs (when a group becomes inactive). The process performing the archiving does not have to wait for the checkpoint of a log switch to complete before it can access the inactive group for archiving.

Figure 22 – 1 illustrate how the database's online redo log is generated by the process archiving the filled groups (ARCH in this illustration).

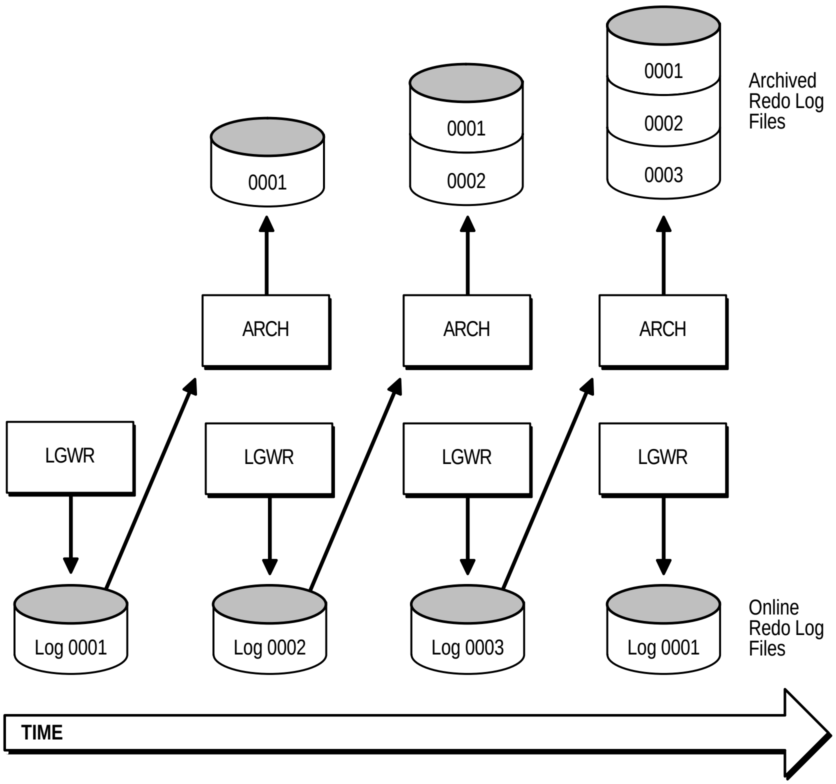


Figure 22 – 1 Online Redo Log File Use in ARCHIVELOG Mode

ARCHIVELOG mode enables complete recovery from disk failure as well as instance failure, because all changes made to the database are permanently saved in an archived redo log.

If *all* databases in a distributed database operate in ARCHIVELOG mode, you can perform coordinated distributed database recovery. However, if *any* database in a distributed database uses NOARCHIVELOG mode, recovery of a global distributed database (to make all databases consistent) is limited by the last full backup of any database operating in NOARCHIVELOG mode.

Also, the entire database can be open and available for normal use while you back up or recover all or part of the database in ARCHIVELOG mode. Note that extra administrative operations are required to manage the files of the archived redo log, and that you must have a dedicated tape drive or additional disk space to store the archived redo log files when the database operates in ARCHIVELOG mode.

You must also decide how filled groups of the online redo log are to be archived. An instance can be configured to have Oracle automatically archive filled online redo log files, or you can manually archive filled groups.

See Also: You can also configure Oracle to verify redo log blocks when they are archived. For more information, see "Verifying Blocks in Redo Log Files" on page 5 – 14.

Turning Archiving On and Off

This section describes aspect of archiving, and includes the following topics:

- Setting the Initial Database Archiving Mode
- Changing the Database Archiving Mode
- Enabling Automatic Archiving
- Disabling Automatic Archiving
- Performing Manual Archiving

You set a database's initial archiving mode as part of database creation. Usually, you can use the default of NOARCHIVELOG mode at database creation because there is no need to archive the redo information generated then. After creating the database, decide whether to change from the initial archiving mode.

After a database has been created, you can switch the database's archiving mode on demand. However, you should generally not switch the database between archiving modes.

See Also: If a database is automatically created during Oracle installation, the initial archiving mode of the database is operating system specific. See your operating system–specific Oracle documentation.

Setting the Initial Database Archiving Mode

When you create the database, you set the initial archiving mode of the redo log in the CREATE DATABASE statement. If you do not specify either ARCHIVELOG or NOARCHIVELOG, NOARCHIVELOG is the default.

See Also: See Chapter 2 for more information about creating a database.

Changing the Database Archiving Mode

To switch a database's archiving mode between NOARCHIVELOG and ARCHIVELOG mode, use the SQL command ALTER DATABASE with the ARCHIVELOG or NOARCHIVELOG option. The following statement switches the database's archiving mode from NOARCHIVELOG to ARCHIVELOG:

```
ALTER DATABASE ARCHIVELOG;
```

Before switching the database's archiving mode, perform the following operations:

To Prepare to Switch Database Archiving Mode

1. Shut down the database instance.

   An open database must be closed and dismounted and any associated instances shut down before the database's archiving mode can be switched. Archiving cannot be disabled if any datafiles need media recovery.

2. Back up the database.

   Before making any major alteration to a database, always back up the database to protect against any problems that might occur.

3. Perform any operating system specific steps *(optional)*.

   These steps may involve exiting Server Manager to configure how Oracle will perform the archiving of the filled groups. Once this operation is complete, start Server Manager again and continue to Step 4.

4. Start up a new instance and mount but do not open the database.

   To enable or disable archiving, the database must be mounted but not open.

   > **Note:** If you are using the Oracle Parallel Server, you must mount the database exclusively, using one instance, to switch the database's archiving mode.

5. Switch the database's archiving mode.

After using the ALTER DATABASE command to switch a database's archiving mode, open the database for normal operation. If you switched to ARCHIVELOG mode, you should also set the archiving options—decide whether to enable Oracle to archive groups of online redo log files automatically as they fill.

See Also: For more information about starting an instance and mounting a database, see Chapter 3.

If you want to archive filled groups, you may have to execute some additional steps at this point, depending on your operating system; see your operating system–specific Oracle documentation for details for your system.

For more information about database backup, see Chapter 23.

See the *Oracle7 Parallel Server Concepts & Administration* guide for more information about switching the archiving mode when using the Oracle Parallel Server.

Enabling Automatic Archiving

If your operating system permits, you can enable automatic archiving of the online redo log. Under this option, no action is required to copy a group after it fills; Oracle automatically archives groups after they are filled. For this convenience alone, automatic archiving is the method of choice for archiving the filled groups of online redo log files.

To enable automatic archiving after instance startup, you must be connected to Oracle with administrator privileges.

☞ **Attention:** Oracle does not automatically archive log files unless the database is also in ARCHIVELOG mode.

Automatic archiving can be enabled before or after instance startup.

See Also: See your operating system–specific Oracle documentation to determine whether this is a valid option for your Oracle Server.

Always specify an archived redo log destination and filename format when enabling automatic archiving; see “Specifying the Archived Redo Log Filename Format and Destination” on page 22 – 11.

If automatic archiving is enabled, manual archiving is still possible; see “Performing Manual Archiving” on page 22 – 7.

Enabling Automatic Archiving at Instance Startup

To enable automatic archiving of filled groups each time an instance is started, include the LOG_ARCHIVE_START parameter, set to TRUE, in the database’s parameter file:

```
LOG_ARCHIVE_START=TRUE
```

The new value takes effect the next time you start the database.

Enabling Automatic Archiving After Instance Startup

To enable automatic archiving of filled online redo log groups without shutting down the current instance, use the SQL command ALTER SYSTEM with the ARCHIVE LOG START parameter; you can optionally include the archiving destination.

The following statement enables archiving:

```
ALTER SYSTEM ARCHIVE LOG START;
```

Using either of the options above, the instance does not have to be shut down to enable automatic archiving. However, if an instance is shut down and restarted after automatic archiving is enabled, the instance is reinitialized using the settings of the parameter file, which may or may not enable automatic archiving.

Disabling Automatic Archiving

You can disable automatic archiving of the online redo log groups at any time. However, once automatic archiving is disabled, you must manually archive groups of online redo log files in a timely fashion. If a database is operated in ARCHIVELOG mode, automatic archiving is disabled, and all groups of online redo log files are filled but not archived, then LGWR cannot reuse any inactive groups of online redo log groups to continue writing redo log entries. Therefore, database operation is temporarily suspended until the necessary archiving is performed.

To disable automatic archiving after instance startup, you must be connected with administrator privilege and have the ALTER SYSTEM privilege.

Automatic archiving can be disabled at or after instance startup.

Disabling Automatic Archiving at Instance Startup

To disable the automatic archiving of filled online redo log groups each time a database instance is started, set the LOG_ARCHIVE_START parameter of a database's parameter file to FALSE:

```
LOG_ARCHIVE_START=FALSE
```

The new value takes effect the next time the database is started.

Disabling Automatic Archiving after Instance Startup

To disable the automatic archiving of filled online redo log groups without shutting down the current instance, use the SQL command ALTER SYSTEM with the ARCHIVE LOG STOP parameter. The following statement stops archiving:

```
ALTER SYSTEM ARCHIVE LOG STOP;
```

If ARCH is archiving a redo log group when you attempt to disable automatic archiving, ARCH finishes archiving the current group, but does not begin archiving the next filled online redo log group.

The instance does not have to be shut down to disable automatic archiving. However, if an instance is shut down and restarted after automatic archiving is disabled, the instance is reinitialized using the settings of the parameter file, which may or may not enable automatic archiving.

Performing Manual Archiving

If a database is operating in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived. You can manually archive groups of the online redo log whether or not automatic archiving is enabled:

- If automatic archiving is not enabled, you must manually archive groups of filled online redo log files in a timely fashion. If all online redo log groups are filled but not archived, LGWR cannot reuse any inactive groups of online redo log members to continue writing redo log entries. Therefore, database operation is temporarily suspended until the necessary archiving is performed.
- If automatic archiving is enabled, but you want to rearchive an inactive group of filled online redo log members to another location, you can use manual archiving. (However, the instance can decide to reuse the redo log group before you have finished manually archiving, and thereby overwrite the files; if this happens, Oracle will put an error message in the ALERT file.)

To manually archive a filled online redo log group, you must be connected with administrator privileges.

Manually archive inactive groups of filled online redo log members using the SQL command ALTER SYSTEM with the ARCHIVE LOG clause.

The following statement archives all unarchived log files:

```
ALTER SYSTEM ARCHIVE LOG ALL;
```

See Also: With both manual or automatic archiving, you need to specify a thread only when you are using the Oracle Parallel Server. See the *Oracle7 Parallel Server Concepts & Administration* guide for more information.

Tuning Archiving

This section describes aspects of tuning the archive process, and includes the following topics:

- Minimizing the Impact on System Performance
- Improving Archiving Speed

For most databases, the archive process has no effect on overall system performance. In some large database sites, however, archiving can have an impact on system performance. On one hand, if the archiver works very quickly, overall system performance can be reduced while the archiver runs, since CPU cycles are being consumed in archiving. On the other hand, if the archiver runs extremely slowly, it has little detrimental effect on system performance, but it takes longer to archive redo log files, and can be a bottleneck if all redo log groups are unavailable because they are waiting to be archived.

For these large database sites, you can tune archiving, to cause it to run either as slowly as possible without being a bottleneck, or as quickly as possible without reducing system performance substantially. To do so, adjust the values of the initialization parameters LOG_ARCHIVE_BUFFERS (the number of buffers allocated to archiving) and LOG_ARCHIVE_BUFFER_SIZE (the size of each such buffer).

> **Note:** When you change the value of LOG_ARCHIVE_BUFFERS or LOG_ARCHIVE_BUFFER_SIZE, the new value takes effect the next time you start the instance.

Minimizing the Impact on System Performance

To make the archiver work as slowly as possible without forcing the system to wait for redo logs, begin by setting the number of archive buffers (LOG_ARCHIVE_BUFFERS) to 1 and the size of each buffer (LOG_ARCHIVE_BUFFER_SIZE) to the maximum possible.

If the performance of the system drops significantly while the archiver is working, make the value of LOG_ARCHIVE_BUFFER_SIZE lower, until system performance is no longer reduced when the archiver runs.

> **Note:** If you want to set archiving to be very slow, but find that Oracle frequently has to wait for redo log files to be archived before they can be reused, consider creating additional redo log file groups. Adding groups can ensure that a group is always available for Oracle to use.

Improving Archiving Speed

To improve archiving performance (for example, if you want to stream input to a tape drive), use multiple archive buffers, so that the archiver process can read the archive log at the same time that it writes the output log. You can set LOG_ARCHIVE_BUFFERS to 2, but for a very fast tape drive you might want to set it to 3 or more. Then, set the size of the archive buffers to a moderate number, and increase it until archiving is as fast as you want it to be without impairing system performance.

See Also: This maximum is operating system dependent; see your operating system–specific Oracle documentation.

For more information about these parameters, see the *Oracle7 Server Reference* guide.

Displaying Archiving Status Information

To list archive status information, you must be connected with administrator privileges.

To see the current archiving mode, query the V$DATABASE view :

```
SELECT log_mode FROM sys.v$database;

LOG_MODE
------------
NOARCHIVELOG
```

The V$ARCHIVE and V$LOG data dictionary views also contain archiving information of a database. For example, the following query lists all log groups for the database and indicates the ones that remain to be archived:

```
SELECT group#, archived
   FROM sys.v$log;

GROUP#     ARC
---------- ---
1 YES
2 NO
```

The command ARCHIVE LOG with the LIST parameter also shows archiving information for the connected instance:

```
ARCHIVE LOG LIST;

Database log mode                        ARCHIVELOG
Automatic archival                       ENABLED
Archive destination                      destination
Oldest online log sequence               30
Next log sequence to archive             32
Current log sequence number              33
```

This display tells you all the necessary information regarding the redo log settings for the current instance:

- The database is currently operating in ARCHIVELOG mode.
- Automatic archiving is enabled.
- The destination of the archived redo log (operating system specific) is *destination* (corresponds to LOG_ARCHIVE_DEST or an overriding destination).
- The oldest filled online redo log group has a sequence number of 30.
- The next filled online redo log group to archive has a sequence number of 32.
- The current online redo log file has a sequence number of 33.

You must archive all redo log groups with a sequence number equal to or greater than the *Next log sequence to archive*, yet less than the *Current log sequence number*. For example, the display above indicates that the online redo log group with sequence number 32 needs to be archived.

Specifying the Archived Redo Log Filename Format and Destination

When the database is used in ARCHIVELOG mode, Oracle must know the archived redo log filename format and destination so that automatic or manual archiving creates uniquely named archived redo log files in the proper location.

Archived redo log files are uniquely named as specified by the LOG_ARCHIVE_FORMAT parameter. Filename format is operating system specific; for most operating systems it consists of a text string, one or more parameters, and a filename extension. When a filled online redo log group is archived, the archiving process concatenates the supplied text string with the return values of the specified parameters to create uniquely identified archived redo log files. Each parameter has an upper bound, which is operating system dependent.

Table 22 – 1 lists the parameters that can be included in a filename format and corresponding examples to show how the parameter affects the filenames created by the archiving process.

Parameter	Description	Example[1]
%T	thread number, left–zero–padded	arch0000000001
%t	thread number, not padded	arch1
%S	log sequence number, left–zero–padded	arch0000000251
%s	log sequence number, not padded	arch251

Table 22 – 1 Archived Redo Log Filename Format Parameters

[1] *Assume LOG_ARCHIVE_FORMAT=arch%parameter, and the upper bound for all parameters is 10 characters.*

The different options are provided so that you can customize the archived redo log filenames as you need. For example, you might want to take into account the operating system sorting algorithm used to list filenames.

The %T and %t are useful only when the Oracle Parallel Server is used. In a non–Parallel Server configuration, you must decide whether to use %S or %s to identify each archived redo log file uniquely. The following is a typical example of a common archived redo log filename format:

```
LOG_ARCHIVE_FORMAT = arch%S.arc
```

Here, *arch* is the filename, %S is the zero–padded log sequence parameter, and *.arc* is the file extension. Assuming the upper bound for the %S parameter is four, this filename format generates archived redo log filenames of the following format:

```
arch0001.arc
arch0002.arc
arch0003.arc
  .
  .
```

Take into account the maximum operating system filename length when specifying the archive filename format. If ARCH or a user process attempts to archive a file and the supplied filename format is too large, the process fails to archive the file.

> **Note:** If no archived filename format is specified using LOG_–ARCHIVE_FORMAT, Oracle uses a default filename format that is operating system–specific.

The archived redo log destination is also operating system–specific. For most operating systems, the archive redo log destination points to a disk drive and a file directory. If permitted by your Oracle Server, this destination can also point to a tape drive dedicated to Oracle for archiving filled online redo log files.

The archived redo log destination is determined at instance startup by the LOG_ARCHIVE_DEST initialization parameter, but can be overridden while the instance is up:

- If a database's parameter file is edited to include a destination using the LOG_ARCHIVE_DEST parameter, the current instance must be shut down and restarted to read the new parameter file.
- If the current instance cannot be shut down, but the archived redo log destination must be specified or changed for automatic archiving, use the ALTER SYSTEM ARCHIVE LOG START 'destination' statement to override the automatic archiving destination.
- During manual archiving, a specified destination overrides the default archived redo log destination. However, automatic archiving continues to use the current automatic archive destination. If no destination is specified, Oracle automatically uses the destination specified by the LOG_ARCHIVE_DEST parameter of the parameter file used to start the instance. If no destination is supplied by the LOG_ARCHIVE_DEST parameter, Oracle uses a default destination that is operating system–dependent.

See Also: See your operating system–specific Oracle documentation for more information about the LOG_ARCHIVE_FORMAT and LOG_ARCHIVE_DEST initialization parameters, and the default archived redo log filename format and destination.

For more information about filename format parameters and the term "thread" see the *Oracle7 Parallel Server Concepts & Administration* guide.

CHAPTER

23

Backing Up a Database

This chapter explains how to back up the data in an Oracle database, and includes the following topics:

- Guidelines for Database Backups
- Creating a Backup Strategy
- Read–Only Tablespaces and Backup
- Performing Backups
- Recovering from an Incomplete Online Tablespace Backup
- Using the Export and Import Utilities for Supplemental Database Protection

See Also: This chapter contains several references to Oracle Server Manager. For more information about performing specific tasks using Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide.*

Guidelines for Database Backups

This section describes guidelines to consider before performing database backups, and includes the following topics:

- Test Backup and Recovery Strategies
- Perform Operating System Backups Frequently and Regularly
- Backup Appropriate Portions of the Database When Making Structural Changes
- Back Up Often–Used Tablespaces Frequently
- Keep Older Backups
- Export Database Data for Added Protection and Flexibility
- Consider Distributed Database Backups
- Back Up after Creating Unrecoverable Objects

Before you create an Oracle database, you should decide how you plan to protect the database against potential disk failures, or to enable point–in–time recovery (if desired). If such planning is not considered before database creation, database recovery may not be possible if a disk failure damages the datafiles, online redo log files, or control files of a database.

See Also: See "Creating a Backup Strategy" on page 23 – 5 if you are not familiar with typical backup strategies for a database.

Test Backup and Recovery Strategies

Test your backup and recovery strategies in a test environment before and after you move to a production system. By doing so, you can test the thoroughness of your strategies and minimize problems before they occur in a real situation.

Performing test recoveries regularly ensures that your archiving, backup, and recovery procedures work. It also helps you stay familiar with recovery procedures, so that you are less likely to make a mistake in a crisis.

Perform Operating System Backups Frequently and Regularly

Frequent and regular full or partial database backups are essential for any recovery scheme. The frequency of backups should be based on the rate or frequency of changes to database data (such as insertions, updates, and deletions of rows in existing tables, and addition of new tables). If a database's data is changed at a high rate, database backup frequency should be proportionally high. Alternatively, if a database is mainly read–only, and updates are issued only infrequently, the database can be backed up less frequently.

Backup Appropriate Portions of the Database When Making Structural Changes

If you make any of the following structural changes, perform a backup of the appropriate portion of your database immediately before and after completing the alteration:

- create or drop a tablespace
- add or rename (relocate) a datafile in an existing tablespace
- add, rename (relocate), or drop an online redo log group or member

Backing up the appropriate portion of the database depends on the archiving mode of the database, as described below:

- If a database is operated in ARCHIVELOG mode, only a control file backup (using the ALTER DATABASE command with the BACKUP CONTROLFILE option) is required before and after a structural alteration. However, you can back up other parts of the database.
- If the database is operated in NOARCHIVELOG mode, a full offline database backup should be taken immediately before and after the modification, including all datafiles, and control files.

Back Up Often–Used Tablespaces Frequently

If a database is operated in ARCHIVELOG mode, it is acceptable to back up the datafiles of an individual tablespace or even a single datafile. This option is useful if a portion of a database is used more extensively than others, such as the SYSTEM tablespace and tablespaces that contain rollback segments. By taking more frequent backups of the extensively used datafiles of a database, you gather more recent copies of the datafiles. As a result, if a disk failure damages the extensively used datafiles, the more recent backup can restore the damaged files. Only a small number of changes to data need to be applied to roll the restored file forward to the time of the failure, or desired point–in–time recovery, thereby reducing database recovery time.

Keep Older Backups

How long you should keep an older database backup depends on the choices you want for database recovery. If you want to recover to a past point–in–time, you need a database backup taken before that point–in–time. For a database operating in NOARCHIVELOG mode, this means a full database backup. For a database operating in ARCHIVELOG mode, this means you should perform a backup of each datafile, taken individually or together, taken any time before the desired recovery point–in–time, and a backup of the associated control file that reflects the database's structure at the point–in–time of recovery.

For added protection, consider keeping two or more backups (and all archive logs that go with these backups) previous to the current backup.

⚠ **Warning:** After opening the database with the RESETLOGS option, existing backups cannot be used for subsequent recovery beyond the time when the logs were reset. You should therefore shutdown the database and make a full offline backup. Doing so will enable recovery of database changes subsequent to using the RESETLOGS option.

See Also: For more information on the Export utility, see the *Oracle7 Server Utilities* guide.

Export Database Data for Added Protection and Flexibility

Because the Oracle Export utility can selectively export specific objects, you might consider exporting portions or all of a database for supplemental protection and flexibility in a database's backup strategy. Database exports are not a substitute for operating system backups and cannot provide the same complete recovery advantages that the built–in functionality of Oracle offers.

Consider Distributed Database Backups

If a database is a node in a distributed database, consider the following guidelines:

- All databases in the distributed database system should be operated in the same archiving mode.
- If the databases in a distributed database system are operating in ARCHIVELOG mode, backups at each node can be performed autonomously (individually, without time coordination).
- If the databases in a distributed database system are operating in NOARCHIVELOG mode, full offline backups must be performed at the same (global) time, to plan for global distributed database recovery. For example, if a database in New York is backed up at midnight EST, the database in San Francisco should be backed up at 9PM PST.

See Also: For more information about distributed database recovery when databases are operating in NOARCHIVELOG mode, see "Coordinate Distributed Recovery" on page 24 – 5.

Back Up after Creating Unrecoverable Objects

If users are creating tables or indexes using the UNRECOVERABLE option, consider taking backups after the objects are created. When tables and indexes are created as UNRECOVERABLE, no redo is logged, and these objects cannot be recovered from existing backups.

See Also: For information about the UNRECOVERABLE option, see the CREATE TABLE...AS SELECT and CREATE INDEX commands in the *Oracle7 Server SQL Reference.*

Creating a Backup Strategy

Before you create an Oracle database, decide how you plan to protect the database against potential failures. Answer the following questions before developing your backup strategy:

- **Is it acceptable to lose any data if a disk failure damages some of the files that constitute a database?** If it is not acceptable to lose any data, the database must be operated in ARCHIVELOG mode, ideally with a multiplexed online redo log. If it is acceptable to lose a limited amount of data if there is a disk failure, you can operate the database in NOARCHIVELOG mode and avoid the extra work required to archive filled online redo log files.
- **Will you ever need to recover to past points–in–time?** If you need to recover to a past point–in–time to correct an erroneous operational or programmatic change to the database, be sure to run in ARCHIVELOG mode and perform control file backups whenever making structural changes. Recovery to a past point–in–time is facilitated by having a backup control file that reflects the database structure at the desired point–in–time.
- **Does the database need to be available at all times (twenty–four hours per day, seven days per week)?** If so, do not operate the database in NOARCHIVELOG mode because the required full database backups, taken while the database is shutdown, cannot be made frequently, if at all. Therefore, high–availability databases always operate in ARCHIVELOG mode to take advantage of online datafile backups.

Backup Strategies in NOARCHIVELOG Mode

If a database is operated in NOARCHIVELOG mode, filled groups of online redo log files are not being archived. Therefore, the only protection against a disk failure is the most recent full backup of the database.

Plan to take full backups regularly, according to the amount of work that you can afford to lose. For example, if you can afford to lose the amount of work accomplished in one week, make a full offline backup once per week. If you can afford to lose only a day’s work, make a full offline backup every day. For large databases with a high amount of activity, it is usually unacceptable to lose work. Therefore, the database should be operated in ARCHIVELOG mode, and the appropriate backup strategies should be used.

Whenever you alter the physical structure of a database operating in NOARCHIVELOG mode, immediately take a full database backup. An immediate full backup protects the new structure of the database not reflected in the previous full backup.

Backup Strategies in ARCHIVELOG Mode

If a database is operating in ARCHIVELOG mode, filled groups of online redo log files are being archived. Therefore, the archived redo log coupled with the online redo log and datafile backups can protect the database from a disk failure, providing for complete recovery from a disk failure to the instant that the failure occurred (or, to the desired past point–in–time). Following are common backup strategies for a database operating in ARCHIVELOG mode:

- When the database is initially created, perform a full offline backup of the entire database. This initial full backup is the foundation of your backups because it provides copies of all datafiles and the control file of the associated database.

  Note: When you perform this initial full backup, make sure that the database is in ARCHIVELOG mode first. Otherwise, the backed up database files will contain the NOARCHIVELOG mode setting.

- Subsequent full backups are not required, and if a database must remain open at all times, full offline backups are not feasible. Instead, you can take partial online backups to update the backups of a database.

- Take online or offline datafile backups to update backed up information for the database (supplementing the full, initial backup). In particular, the datafiles of extensively used tablespaces should be backed up frequently to reduce database recovery time, should recovery ever be required. If a more recent datafile backup restores a damaged datafile, fewer archived redo logs need to be applied to the restored datafile to roll it forward to the time of the failure.

  Whether you should take online or offline datafile backups depends on the availability requirements of the data. Online datafile backups are the only choice if the data being backed up must always be available.

- Every time you make a structural change to the database, take a control file backup, using the ALTER DATABASE command with the BACKUP CONTROLFILE option.

⚠ **Warning:** If the control file does not contain the name of a datafile, and you have no backup of that datafile, you cannot recover the file if it is lost. Also, do not use operating system utilities to backup the control file in ARCHIVELOG mode, unless you are performing a full, offline backup.

- If you want a copy of the current online log, then archive it. Archiving means the online log will no longer be the current log. If you end up copying a current online log, the copy will appear as the end of the redo thread. However, additional redo may have been generated in the thread. If you ever attempt to execute recovery supplying the redo log copy, recovery will erroneously detect the end of the redo thread and prematurely terminate, possibly corrupting the database.

Read–Only Tablespaces and Backup

You can create backups of a read–only tablespace while the database is open. Immediately after making a tablespace read–only, you should back up the tablespace. Provided the tablespace remains read–only, there is no need to perform any further backups of it.

Unlike backups of writeable tablespaces, you do not need to use the BEGIN and END BACKUP commands to mark the beginning and end of the online backup of a read–only tablespace. Using these commands with reference to a read–only tablespace causes an error.

After you change a read–only tablespace to a read–write tablespace, you need to resume your normal backups of the tablespace, just as you do when you bring an offline read–write tablespace back online.

Bringing the datafiles of a read–only tablespace online does not make these files writeable, nor does it cause the file header to be updated. Thus, it is not necessary to perform a backup of these files, as is necessary when you bring a writeable datafile back online.

Performing Backups

This section describes the various aspects of taking database backups, and includes the following topics:

- Listing Database Files Before Backup
- Performing Full Offline Backups
- Performing Partial Backups
- Performing Control File Backups

Listing Database Files Before Backup

Before taking a full or partial database backup, identify the files to be backed up. Obtain a list of datafiles by querying the V$DATAFILE view:

```
SELECT name FROM v$datafile;
```

Then obtain a list of online redo log files for a database using the query below:

```
SELECT member FROM v$logfile;
```

These queries list the datafiles and online redo log files of a database, respectively, according to the information in the current control file of the database.

Finally, obtain the names of the current control files of the database by issuing the following statement within Server Manager:

```
SHOW PARAMETER control_files;
```

Whenever you take a control file backup (using the ALTER DATABASE command with the BACKUP CONTROLFILE TO 'filename' option), save a list of all datafiles and online redo log files with the control file backup. To obtain this list use the ALTER DATABASE command with the BACKUP CONTROLFILE TO TRACE option. By saving the control file backup with the output of the TO TRACE invocation, the database's physical structure at the time of the control file backup is clearly documented.

Performing Full Offline Backups

Take a full offline backup of all files that constitute a database after the database is shut down to system–wide use in normal priority. *A full backup taken while the database is open, after an instance crash or shutdown abort is useless.* In such cases, the backup is not a full offline backup because the files are inconsistent with respect to the current point–in–time. The files that constitute the database are the datafiles, online redo log files, and control file.

Full offline backups do not require the database to be operated in a specific archiving mode. A full offline backup can be taken if a database is operating in either ARCHIVELOG or NOARCHIVELOG mode.

The set of backup files that result from a full offline backup are consistent. All files correspond to the same point in time. If database recovery is necessary, these files can completely restore the database to an exact point in time. After restoring the backup files, additional recovery steps may be possible to restore the database to a more current time if the database is operated in ARCHIVELOG mode and online redo logs are *not* restored.

Warning: A backup control file created during a full database backup should only be used with the other files taken in that backup, to restore the full backup. It should not be used for complete or incomplete database recovery. Unless you are taking a full database backup, you should back up your control file using the ALTER DATABASE command with the BACKUP CONTROLFILE option.

See Also: For more information about backing up control files, see “Control File Backups” on page 23 – 14.

Preparing to Take a Full Backup

To guarantee that a database’s datafiles are consistent, always shut down the database with normal or immediate priority before making a full database backup. Never perform a full database backup after an instance failure or after the database is shut down with abort priority (that is, using a SHUTDOWN ABORT statement). In this case, the datafiles are probably not consistent with respect to a specific point–in–time.

To Perform a Full Backup

1. Shut down the database with normal or immediate priority.

   To make a full backup, all database files must be closed by shutting down the database. Do not make a full backup when the instance is aborted or stopped because of a failure. Reopen the database and shut it down cleanly before making a full backup.

2. Back up all files that constitute the database.

   Use operating system commands or a backup utility to make backups of all datafiles, online redo log files, and a single control file of the database. If you are multiplexing the online redo log, back up all members of each group, because it is not guaranteed that any one member of a group is complete. Also back up the parameter files associated with the database.

   Operating system backups can be performed:

   - within Server Manager, using the HOST command
   - outside Server Manager, with the operating system commands or a backup utility

3. Restart the database.

   After you have finished backing up all datafiles, online redo log files, and a single control file of the database, you can restart the database.

Verifying Backups

DB_VERIFY is a command–line utility that performs a physical data structure integrity check on database files. Use DB_VERIFY primarily when you need to ensure that a backup database (or datafile) is valid before it is restored.

See Also: See page 3 – 8 for more information about database shutdown.

For more information about making operating system backups of files, see your operating system–specific Oracle documentation.

For more information on DB_VERIFY, see the *Oracle7 Server Utilities* guide.

Performing Partial Backups

You can perform different types of partial backups:

- online tablespace and datafile backups
- offline tablespace and datafile backups

Partial backups should only be taken (and in some cases *can* only be taken) if a database is operating in ARCHIVELOG mode. Partial backups cannot be used to restore a database operating in NOARCHIVELOG mode.

Online Tablespace and Datafile Backups

All datafiles of an individual online tablespace or specific datafiles of an online tablespace can be backed up while the tablespace and datafiles are currently online and in use for normal database operation.

To back up online tablespaces, you must have the MANAGE TABLESPACE system privilege.

To Perform an Online Backup of an Entire Tablespace or Specific Datafile

1. Identify the datafiles.

   If you are backing up a specific datafile, use the fully specified filename of the datafile.

   Before beginning a backup on an entire tablespace, identify all of the tablespace’s datafiles using the DBA_DATA_FILES data dictionary view. For example, assume that the USERS tablespace is to be backed up. To identify the USERS tablespace’s datafile, you can query the DBA_DATA_FILES view:

```
SELECT tablespace_name, file_name
   FROM sys.dba_data_files
   WHERE tablespace_name = ’USERS’;

TABLESPACE_NAME    FILE_NAME
---------------    ---------
```

```
USERS              filename1
USERS              filename2
```

Here, *filename1* and *filename2* are fully specified filenames corresponding to the datafiles of the USERS tablespace.

2. Mark the beginning of the online tablespace backup.

   To prepare the datafiles of an online tablespace for backup, use either the Start Online Backup menu item of Server Manager, or the SQL command ALTER TABLESPACE with the BEGIN BACKUP option.

   The following statement marks the start of an online backup for the tablespace USERS:

   ```
   ALTER TABLESPACE users BEGIN BACKUP;
   ```

   ⚠ **Warning:** If you forget to mark the beginning of an online tablespace backup, or neglect to assure that the BEGIN BACKUP command has completed before backing up an online tablespace, the backup datafiles are not useful for subsequent recovery operations. Attempting to recover such a backup is a risky procedure, and can return errors that result in inconsistent data later. For example, the attempted recovery operation will issue a "fuzzy files" warning, and lead to an inconsistent database that will not open.

3. Back up the online datafiles.

   At this point, you can back up the online datafiles of the online tablespace from within Server Manager, using the HOST command, by exiting Server Manager and entering the operating system commands, or starting the Backup utility

4. Mark the end of the online tablespace backup.

   After backing up the datafiles of the online tablespace, indicate the end of the online backup using either the End Online Tablespace Backup dialog box of Server Manager, or the SQL command ALTER TABLESPACE with the END BACKUP option.

The following statement ends the online backup of the tablespace USERS:

```
ALTER TABLESPACE users END BACKUP;
```

If you forget to indicate the end of an online tablespace backup, and an instance failure or SHUTDOWN ABORT occurs, Oracle assumes that media recovery (possibly requiring archived redo logs) is necessary at the next instance start up.

See Also: See the *Oracle7 Server Reference* for more information about the DBA_DATA_FILES data dictionary view.

See your operating system–specific Oracle documentation for more information about making operating system backups of files.

To restart the database without media recovery, see "Recovering From an Incomplete Online Tablespace Backup" on page 23 – 17.

Determining Datafile Backup Status To view the backup status of a datafile, you can use the data dictionary table V$BACKUP. This table lists all online files and gives their backup status. It is most useful when the database is open. It is also useful immediately after a crash, because it shows the backup status of the files at the time of the crash. You can use this information to determine whether you have left tablespaces in backup mode.

> **Note:** V$BACKUP is not useful if the control file currently in use is a restored backup or a new control file created since the media failure occurred. A restored or re–created control file does not contain the information Oracle needs to fill V$BACKUP accurately. Also, if you have restored a backup of a file, that file's STATUS in V$BACKUP reflects the backup status of the older version of the file, not the most current version. Thus, this view might contain misleading information on restored files.

For example, the following query displays the current backup status of datafiles:

```
SELECT file#, status
   FROM v$backup;

FILE#       STATUS
---------------------
   0011     INACTIVE
   0012     INACTIVE
   0013     ACTIVE
...
```

In the STATUS column, “INACTIVE” indicates that the file is not currently being backed up. “ACTIVE” indicates that the file is marked as currently being backed up.

Backing Up Several Online Tablespaces If you have to back up several online tablespaces, use either of the following procedures:

- Back up the online tablespaces in parallel. For example, prepare all online tablespaces for backup:

  ```
  ALTER TABLESPACE ts1 BEGIN BACKUP;
  ALTER TABLESPACE ts2 BEGIN BACKUP;
  ALTER TABLESPACE ts3 BEGIN BACKUP;
  ```

  Next, back up all files of the online tablespaces and indicate that the online backups have been completed:

  ```
  ALTER TABLESPACE ts1 END BACKUP;
  ALTER TABLESPACE ts2 END BACKUP;
  ALTER TABLESPACE ts3 END BACKUP;
  ```

- Back up the online tablespaces serially. For example, individually prepare, back up, and end the backup of each online tablespace:

  ```
  ALTER TABLESPACE ts1 BEGIN BACKUP;
  backup files
  ALTER TABLESPACE ts1 END BACKUP;
  ALTER TABLESPACE ts2 BEGIN BACKUP;
  backup files
  ALTER TABLESPACE ts2 END BACKUP;
  ```

The second option minimizes the time between ALTER TABLESPACE... BEGIN / END BACKUP commands and is recommended. During online backups, more redo information is generated for the tablespace.

Offline Tablespace and Datafile Backups

All or some of the datafiles of an individual tablespace can be backed up while the tablespace is offline. All other tablespaces of the database can remain open and available for system–wide use.

> **Note:** You cannot take the SYSTEM tablespace or any tablespace with active rollback segments offline. The following procedure cannot be used for such tablespaces.

To take tablespaces offline and online, you must have the MANAGE TABLESPACE system privilege.

To Back Up the Offline Datafiles of an Offline Tablespace

1. Identify the datafiles of the offline tablespace.

   Use the fully specified filename of the datafile.

   Before taking the tablespace offline, identify the names of its datafiles by querying the data dictionary view DBA_DATA_FILES. (See Step 1 on page 23 – 10.)

2. Take the tablespace offline, using normal priority if possible.

   Use of normal priority, if possible, is recommended because it guarantees that the tablespace can be subsequently brought online without the requirement for tablespace recovery.

   To take a tablespace and all associated datafiles offline with normal priority, use the Take Offline menu item of Server Manager, or the SQL command ALTER TABLESPACE with the OFFLINE parameter. The following statement takes a tablespace named USERS offline normally:

   ```
   ALTER TABLESPACE users OFFLINE NORMAL;
   ```

   After a tablespace is taken offline with normal priority, all datafiles of the tablespace are closed.

3. Back up the offline datafiles.

   At this point, you can back up the datafiles of the offline tablespace from within Server Manager using the HOST command, by exiting Server Manager and entering the operating system commands, or starting the Backup utility.

4. Bring the tablespace online. *(Optional)*

   Bring the tablespace online using either the Place Online menu item of Server Manager, or the SQL command ALTER TABLESPACE with the ONLINE option. The following statement brings an offline tablespace named USERS online:

```
ALTER TABLESPACE users ONLINE;
```

> **Note:** If you took the tablespace offline using temporary or immediate priority, the tablespace may not be brought online unless tablespace recovery is performed.

After a tablespace is brought online, the datafiles of the tablespace are open and available for use.

See Also: For more information about online and offline tablespaces, see page 8 – 7.

For more information about making operating system backups of files, see your operating system–specific Oracle documentation.

For more information about tablespace recovery, see page 24 – 20.

Performing Control File Backups

Back up the control file of a database after making a structural modification to a database operating in ARCHIVELOG mode.

To backup a database's control file, you must have the ALTER DATABASE system privilege.

You can take a backup of a database's control file using the SQL command ALTER DATABASE with the BACKUP CONTROLFILE option. The following statement backs up a database's control file:

```
ALTER DATABASE BACKUP CONTROLFILE TO 'filename' REUSE;
```

Here, *filename* is a fully specified filename that indicates the name of the new control file backup.

The REUSE option allows you to have the new control file overwrite a control file that currently exists.

Backing Up the Control File to the Trace File

The TRACE option of the ALTER DATABASE BACKUP CONTROLFILE command helps you manage and recover your control file. TRACE prompts Oracle to write SQL commands to the database's trace file, rather than making a physical backup of the control file. These commands start up the database, re–create the control file, and recover and open the database appropriately, based on the current control file. Each command is commented. Thus, you can copy the commands from the trace file into a script file, edit them as necessary, and use the script to recover the database if all copies of the control file are lost (or to change the size of the control file).

For example, assume the SALES database has three enabled threads, of which thread 2 is public and thread 3 is private. It also has multiplexed redo log files, and one offline and one online tablespace.

```
ALTER DATABASE
   BACKUP CONTROLFILE TO TRACE NORESETLOGS;

3–JUN–1992 17:54:47.27:
# The following commands will create a new control file and use it
# to open the database.
# No data other than log history will be lost. Additional logs may
# be required for media recovery of offline data files. Use this
# only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE SALES NORESETLOGS ARCHIVELOG
   MAXLOGFILES 32
   MAXLOGMEMBERS 2
   MAXDATAFILES 32
   MAXINSTANCES 16
   MAXLOGHISTORY 1600
LOGFILE
   GROUP 1
       ’/diska/prod/sales/db/log1t1.dbf’,
       ’/diskb/prod/sales/db/log1t2.dbf’
   )  SIZE 100K
   GROUP 2
       ’/diska/prod/sales/db/log2t1.dbf’,
       ’/diskb/prod/sales/db/log2t2.dbf’
   ) SIZE 100K,
   GROUP 3
        ’/diska/prod/sales/db/log3t1.dbf’,
        ’/diskb/prod/sales/db/log3t2.dbf’
   ) SIZE 100K
DATAFILE
   ’/diska/prod/sales/db/database1.dbf’,
   ’/diskb/prod/sales/db/filea.dbf’
;
# Take files offline to match current control file.
ALTER DATABASE DATAFILE ’/diska/prod/sales/db/filea.dbf’ OFFLINE

# Recovery is required if any data files are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE;

# All logs need archiving and a log switch is needed.
ALTER SYSTEM ARCHIVE LOG ALL;

# Database can now be opened normally
ALTER DATABASE OPEN;
```

```
#  Files in normal offline tablespaces are now named.
ALTER DATABASE RENAME FILE 'MISSING0002'
   TO '/diska/prod/sales/db/fileb.dbf';
```

Using the command without NORESETLOGS produces the same output. Using the command with RESETLOGS produces a similar script that includes commands that recover and open the database, but resets the redo logs upon startup.

Recovering From an Incomplete Online Tablespace Backup

The following situations can cause an incomplete tablespace backup:

- You did not indicate the end of the online tablespace backup operation (using the ALTER TABLESPACE command with the END BACKUP option), and the database was subsequently shut down with the ABORT option.
- A system or instance failure, or SHUTDOWN...ABORT interrupted the backup.

Upon detecting an incomplete online tablespace backup at startup, Oracle assumes that media recovery (possibly requiring archived redo log) is necessary for startup to proceed. You can avoid performing media recovery by using the ALTER DATABASE DATAFILE...END BACKUP command. Remember to list all the datafiles of the tablespaces that were in the process of being backup up before the database was restarted. You can determine whether datafiles were in the process of being backed up by querying the V$BACKUP view.

⚠ **Warning:** Do not use ALTER DATABASE DATAFILE...END BACKUP if you have restored any of the affected files from a backup.

After you have restarted your database, you can perform the recovery in either of two ways:

- Use the STARTUP RECOVER command to start and recover the database automatically.
- Start an instance, open and mount the database, and issue the statement RECOVER DATABASE.

The first method is easier because it prompts Oracle to perform recovery only if it is needed.

See Also: For information on starting the database, see page 3 – 2.

For information on recovering a database, see page 24 – 7.

Using the Export and Import Utilities for Supplemental Database Protection

This section describes the Import and Export utilities, and includes the following topics:

- Using Import
- Using Export

Export and Import are utilities that move Oracle data in and out of Oracle databases. Export writes data from an Oracle database to an operating system file in a special format. Import reads Export files and restores the corresponding information into an existing database. Although Export and Import are designed for moving Oracle data, you can also use them to supplement backups of data.

See Also: Both the Export and Import utilities are described in detail in the *Oracle7 Server Utilities* guide.

Using Export

The Export utility allows you to backup your database while it is open and available for use. It writes a read–consistent view of the database's objects to an operating system file. System audit options are not exported.

⚠ **Warning:** If you use Export to backup, all data must be exported in a logically consistent way so that the backup reflects a single point in time. No one should make changes to the database while the Export takes place. Ideally, you should run the database in restricted mode while you export the data, so no regular users can access the data.

Table 23 – 1 lists available export modes.

Mode	Description
User	exports all objects owned by a user
Table	exports all or specific tables owned by a user
Full Database	exports all objects of the database

Table 23 – 1 Export Modes

Following are descriptions of Export types:

Incremental Export — Only database data that has changed since the last incremental, cumulative, or complete export is exported. An incremental export exports the object's definition and all its data. Incremental exports are typically performed more often than cumulative or complete reports.

For example, if tables A, B, and C exist, and only table A's information has been modified since the last incremental export, only table A is exported.

Cumulative Exports

Only database data that has been changed since the last cumulative or complete export is exported.

Perform this type of export on a limited basis, such as once a week, to condense the information contained in numerous incremental exports.

For example, if tables A, B, and C exist, and only table A's and table B's information has been modified since the last cumulative export, only the changes to tables A and B are exported.

Complete Exports

All database data is exported.

Perform this type of export on a limited basis, such as once a month, to export all data contained in a database.

Using Import

The Import utility allows you to restore the database information held in previously created Export files. It is the complement utility to Export.

To recover a database using Export files and the Import utility:

- Re–create the database structure, including all tablespaces and users

  Note: These re–created structures should not have objects in them.

- Import the appropriate Export files to restore the database to the most current state possible. Depending on how your Export schedule is performed, imports of varying degrees will be necessary to restore a database.

Assume that the schedule illustrated in Figure 23 – 1 is used in exporting data from an Oracle database.

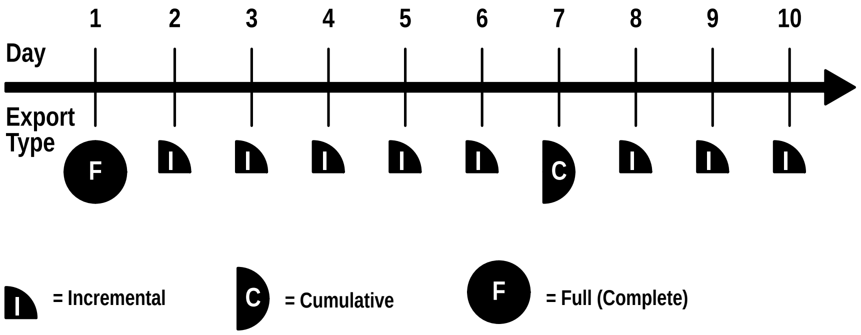


Figure 23 – 1 A Typical Export Schedule

A complete export was taken on Day 1, a cumulative export was taken every week, and incremental exports were taken daily.

To recover from a disk failure that occurs on Day 10, before the next incremental export is taken on Day 11

1. Recreate the database, including all tablespaces and users.
2. Import the complete database export taken on Day 1.
3. Import the cumulative database export taken on Day 7.
4. Import the incremental database exports taken on Days 8, 9, and 10.

CHAPTER

24 Recovering a Database

This chapter describes how to recover a database, and includes the following topics:

- Fundamental Recovery Concepts and Strategies
- Preparing for Media Recovery
- Performing Complete Media Recovery
- Performing Incomplete Media Recovery
- Planning and Preparing for Disaster Recovery
- Unrecoverable Objects
- Read–Only Tablespaces and Recovery
- Recovery Procedure Examples

See Also: Occasionally, this chapter refers you to Oracle Server Manager. To learn how to use Server Manager / GUI or Server Manager / LineMode, see the *Oracle Server Manager User's Guide*.

Fundamental Recovery Concepts and Strategies

Before recovering a database, familiarize yourself with the fundamental data structures, concepts and strategies of Oracle recovery. This section describes basic recovery issues, and includes the following topics:

- Important Recovery Data Structures
- Recovery Operations
- Recovery Planning and Strategies

Important Recovery Data Structures

Table 24 – 1 describes important data structures involved in recovery processes. Be familiar with these data structures before starting any recovery procedure.

Data Structure	Description
Control File	The control file contains records that describe and maintain information about the physical structure of a database. The control file is updated continuously during database use, and must be available for writing whenever the database is open. If the control file is not accessible, the database will not function properly.
System Change Number (SCN)	The system change number is a clock value for the Oracle database that describes a committed version of the database. The SCN functions as a sequence generator for a database, and controls concurrency and redo record ordering. Think of the SCN as a timestamp that helps ensure transaction consistency.
Redo Records	A redo record is a group of change vectors describing a single, atomic change to the database. Redo records are constructed for all data block changes and saved on disk in the redo log. Redo records allow multiple database blocks to be changed so that either all changes occur or no changes occur, despite arbitrary failures.
Redo Logs	All changes to the Oracle database are recorded in redo logs, which consist of at least two redo log files that are separate from the datafiles. During database recovery from an instance or media failure, Oracle applies the appropriate changes in the database's redo log to the datafiles; this updates database data to the instant that the failure occurred.
Rollback Segments	Information in a rollback segment is used during database recovery to undo any uncommitted changes applied from the redo log to the datafiles. After the rollback segments are used to remove all uncommitted data from the datafiles, data is in a consistent state.
Backup	A database backup consists of operating system backups of the physical files that constitute the Oracle database. To begin database recovery from a media failure, Oracle uses file backups to restore damaged datafiles or control files.

Table 24 – 1 Important Recovery Data Structures

Checkpoint	A checkpoint is a data structure in the control file that defines a consistent point of the database across all threads of a redo log. Checkpoints are similar to SCNs, and also describe which threads exist at that SCN. Checkpoints are used by recovery to ensure that Oracle starts reading the log threads for the redo application at the correct point. For Parallel Server, each checkpoint has its own redo information.

Table 24 – 1 Important Recovery Data Structures

See Also: For more information about these and other data structures, see the *Oracle7 Server Concepts* manual.

Recovery Operations

Media recovery restores a database's datafiles to the most recent point–in–time before disk failure, and includes the committed data in memory that was lost due to failure. Following is a list of media recovery operations:

- Media Recovery
  1. Complete Media Recovery
     - Closed Database Recovery
     - Open–Database, Offline–Tablespace Recovery
     - Open–Database, Offline–Tablespace, Individual Datafile Recovery
  2. Incomplete Media Recovery
     - Cancel–Based Recovery
     - Time–Based Recovery
     - Change–Based Recovery

Recovery Planning and Strategies

Before recovering a database, you should create a recovery plan or strategy. This section describes important issues to consider when defining your plan.

Test Backup and Recovery Strategies

You should test your backup and recovery strategies in a test environment before moving to a production system. You should continue to test your system regularly. That way, you can test the thoroughness of your strategies and later avoid real–life crises. Performing test recoveries regularly ensures that your archiving and backup procedures work. It also keeps you familiar with recovery procedures, so that you are less likely to make mistakes in a crisis.

Determine What Type of Recovery Operation Is Appropriate

You can use the RECOVER command when faced with any of the following problems:

- Media failure has damaged your database.

- You need to recover to a point–in–time in the past (for example, undo an erroneous operational or programmatic change to the database).
- You have lost online logs.

Before recovering a database, you must choose an appropriate recovery operation. Your answers to the following questions will determine the most appropriate operation.

1. What recovery operations are available?

   The answer to this first question depends on whether your database is archiving redo logs.

   - If the database is in ARCHIVELOG mode, several recovery operations are available.
   - If the database is in NOARCHIVELOG mode, usually only one recovery operation is available, which is to restore the most recent full backup and re–enter all work performed since the backup. (If you have used Export to supplement regular backups, you can instead use Import to restore data.) Some special losses are easier to repair.

2. What recovery operations are appropriate for this particular problem?

   If the database is in ARCHIVELOG mode, several recovery operations are available to restore a damaged database to a transaction–consistent state.

3. Is the damaged database part of a distributed database?

   If so, database recovery may need to be coordinated among the nodes of the distributed database.

4. Are disaster recovery procedures in place?

   If you have lost all of your online media, or have determined that your recovery time will be too long, you may want to activate your standby database rather than perform media recovery on your primary database.

See Also: For a detailed list of different problems that media failures can cause and the appropriate recovery operations, see page 24 – 47.

Moving Datafiles

The goal of database recovery is to reopen a database for normal operation as soon as possible. If a media failure occurs because of a hardware problem, the damage should be repaired as soon as possible. However, database recovery does not depend on the resolution of long–lasting hardware problems. Table 24 – 2 lists sections in this Guide that contain procedures for restoring files from a damaged device to other storage devices.

Type of File	Section Name	Page
Datafile	Renaming and Relocating Datafiles for Tablespace	9 – 8
Online Redo Log File	Renaming and Relocating Online Redo Log Members	5 – 6
Control File	Creating Additional Copies of the Control File, and Renaming or Relocating Control Files	6 – 4

Table 24 – 2 Damaged File Restoration

Coordinate Distributed Recovery

The Oracle distributed database architecture is autonomous in nature. Therefore, depending on the type of recovery operation selected for a single, damaged database, recovery operations may, or may not, have to be coordinated globally among all databases in the distributed database system. Table 24 – 3 summarizes the different types of recovery operations and whether coordination among nodes of a distributed database system is required.

Type of Recovery Operation	Implication for Distributed Database System
Restoring a full backup for a database that was never accessed (updated or queried) from a remote node	Use non–coordinated, autonomous database recovery.
Restoring a full backup for a database that was accessed by a remote node	Shut down all databases and restore them using the same coordinated full backup.
Complete media recovery of one or more databases in a distributed database	Use non–coordinated, autonomous database recovery.
Incomplete media recovery of a database that was never accessed by a remote node	Use non–coordinated, autonomous database recovery.
Incomplete media recovery of a database that was accessed by a remote node	Use coordinated, incomplete media recovery to the same global point–in–time for all databases in the distributed database.

Table 24 – 3 Database Recovery in a Distributed Database System

Coordinate Time–Based and Change–Based Distributed Database Recovery In special circumstances, one node in a distributed database may require recovery to a past point–in–time. To preserve global data consistency, it is often necessary to recover all other nodes in the system to the same point–in–time. This is called “coordinated, time–based, distributed database recovery.” The following tasks should be performed with the standard procedures of time–based and change–based recovery described in this chapter.

To Coordinate Time–Based, Distributed Recovery Among Many Nodes in a Distributed Database System

1. Recover the database that requires the recovery operation using time–based recovery. For example, if a database needs to be recovered because of a user error (such as an accidental table drop), recover this database first using time–based recovery. Do not recover the other databases at this point.
2. After you have recovered the database and opened it using the RESETLOGS option, look in the ALERT file of the database for the RESETLOGS message.

   If the message is, "RESETLOGS after complete recovery through change nnnnnnnn," you have applied all the changes in the database and performed a complete recovery. Do not recover any of the other databases in the distributed system, or you will unnecessarily remove changes in them. Recovery is complete.

   If the reset message is, "RESETLOGS after incomplete recovery UNTIL CHANGE nnnnnnnn," you have successfully performed an incomplete recovery. Record the change number from the message and proceed to the next step.
3. Recover all other databases in the distributed database system using change–based recovery, specifying the change number (SCN) from Step 2.

Recover Database with Snapshots If a master database is independently recovered to a past point in time (that is, coordinated, time–based distributed database recovery is not performed), any dependent remote snapshot that was refreshed in the interval of lost time will be inconsistent with its master table. In this case, the administrator of the master database should instruct the remote administrators to perform a complete refresh of any inconsistent snapshot.

Preparing for Media Recovery

This section describes issues related to media recovery preparation, and includes the following topics:

- Media Recovery Commands
- Issues Common to All Media Recovery Operations
- Restoring a Full Backup, NOARCHIVELOG Mode
- Specifying Parallel Recovery

See Also: For information about the appropriate method of recovery for each type of problem, see "Examples of Recovery Procedures" on page 24 – 47.

Media Recovery Commands

There are three basic media recovery commands, which differ only in the way the set of files being recovered is determined. They all use the same criteria for determining if files can be recovered. Media recovery signals an error if it cannot get the lock for a file it is attempting to recover. This prevents two recovery sessions from recovering the same file. It also prevents media recovery of a file that is in use. You should be familiar with all media recovery commands before performing media recovery.

RECOVER DATABASE Command

RECOVER DATABASE performs media recovery on all online datafiles that require redo to be applied. If all instances were cleanly shutdown, and no backups were restored, RECOVER DATABASE indicates a no recovery required error. It also fails if any instances have the database open (since they have the datafile locks). To perform media recovery on an entire database (all tablespaces), the database must be mounted EXCLUSIVE and closed.

RECOVER TABLESPACE Command

RECOVER TABLESPACE performs media recovery on all datafiles in the tablespaces listed. To translate the tablespace names into datafile names, the database must be mounted and open. The tablespaces must be offline to perform the recovery. An error is indicated if none of the files require recovery.

RECOVER DATAFILE Command

RECOVER DATAFILE lists the datafiles to be recovered. The database can be open or closed, provided the media recovery locks can be acquired. If the database is open in any instance, then datafile recovery can only recover offline files.

See Also: For more information about recovery commands, see the *Oracle7 Server SQL Reference* guide.

Issues Common to All Media Recovery Operations

This section describes topics common to all complete and incomplete media recovery operations. You should be familiar with these topics before proceeding with any recovery process.

Determining Which Files to Recover

You can often use the table V$RECOVER_FILE to determine which files to recover. This table lists all files that need to be recovered, and explains why they need to be recovered.

> **Note:** The table is not useful if the control file currently in use is a restored backup or a new control file created since the media failure occurred. A restored or re–created control file does not contain the information Oracle needs to fill V$RECOVER_FILE accurately.

The following query displays the file ID numbers of datafiles that require recovery:

```
SELECT file#, online, error
   FROM v$recover_file;

FILE#       ONLINE     ERROR
-----------------------------------------------------
   0014     ONLINE
   0018     ONLINE     FILE NOT FOUND
   0032     OFFLINE    OFFLINE NORMAL
...
```

Use the data dictionary view V$DATAFILE, which contains the file's NAME and FILE#, to find the name of a file based on its file number.

Restoring Damaged Datafiles

If a media failure permanently damages one or more datafiles of a database, you must restore backups of the damaged datafiles before you can recover the damaged files.

Relocating Damaged Files If a damaged datafile cannot be restored to its original location (for example, a disk must be replaced, so the files are restored to an alternate disk), the new locations of these files must be indicated to the control file of the associated database. Therefore, use the procedure given in "Renaming and Relocating Datafiles" on page 9 – 7.

Recovering a Datafile Without a Backup If a datafile is damaged and no backup of the file is available, the datafile can still be recovered if:

- all log files written since the creation of the original datafile are available
- the control file contains the name of the damaged file (that is, the control file is current, or is a backup taken after the damaged datafile was added to the database)

Use the CREATE DATAFILE clause of the ALTER DATABASE command to create a new, empty datafile, replacing a damaged datafile that has no corresponding backup. However, you cannot create a new file based on the first datafile of the SYSTEM tablespace because it contains information not covered by redo logs. For example, assume that the datafile "disk1:users1" has been damaged, and no backup is available. The following statement re–creates the original datafile (same size) on disk 2:

```
ALTER DATABASE CREATE DATAFILE 'disk1:users1' AS 'disk2:users1';
```

> **Note:** The old datafile is renamed as the new datafile when an ALTER DATABASE CREATE DATAFILE statement is executed.

This statement enables you to create an empty file that matches the lost file. Oracle looks at information in the control file and the data dictionary to obtain size information. Next, you must perform media recovery on the empty datafile. All archived redo logs written since the original datafile was created must be mounted and reapplied to the new, empty version of the lost datafile during recovery. If the database was created in NOARCHIVELOG mode, the original datafiles of the SYSTEM tablespace cannot be restored using an ALTER DATABASE CREATE DATAFILE statement because the necessary archived redo logs are not available.

Restoring Necessary Archived Redo Log Files

All archived redo log files required for the pending media recovery eventually need to be on disk, so that they are readily available to Oracle.

To determine which archived redo log files you need, you can use the tables V$LOG_HISTORY and V$RECOVERY_LOG. V$LOG_HISTORY lists all of the archived logs, including their probable names, given the current archived log file naming scheme (as set by the parameter LOG_ARCHIVE_FORMAT). V$RECOVERY_LOG lists only the archived redo logs that Oracle needs to perform recovery. It also includes the probable names of the files, using LOG_ARCHIVE_FORMAT. Be aware that you will need all the redo information from the time the datafile was added to the database.

If space is available, restore all of the required archived redo log files to the location currently specified by the initialization parameter LOG_ARCHIVE_DEST. By doing this, you enable Oracle to locate automatically the correct archived redo log file when required during media recovery. If sufficient space is not available at the location indicated by LOG_ARCHIVE_DEST, you can restore some or all of the required archived redo log files to any disk accessible to Oracle. In this case, you can specify the location of the archived redo log files before or during media recovery.

After an archived log is applied, you can delete the restored copy of the archived redo log file to free disk space. However, make sure that a copy of each archived log group still exists on offline storage.

See Also: For more information about tables, see the *Oracle7 Server Reference*.

Starting Media Recovery

If a damaged database is in ARCHIVELOG mode, it is a candidate for either complete media recovery or incomplete media recovery operations. To begin media recovery operations, use one of the following options of Server Manager:

- the Apply Recovery Archives dialog box
- the Server Manager RECOVER command
- the SQL command ALTER DATABASE

To start any type of media recovery, you must have administrator privileges. All recovery sessions must be compatible. One session cannot start complete media recovery while another performs incomplete media recovery. Also, you cannot start media recovery if you are connected to the database via a multi–threaded server process.

See Also: For more information on multi–threaded server processes, see page 4 – 3.

Recovery Scenarios

The following scenarios describe various ways to invoke media recovery.

Recovering a Closed Database After the database is mounted, but closed, start closed database recovery (complete or incomplete) using either Server Manager's Apply Recovery Archives dialog box, or the RECOVER command with the DATABASE parameter.

The following statement recovers the database up to a specified time using a control file backup:

```
RECOVER DATABASE
   UNTIL TIME '1992–12–31:12:47:30' USING BACKUP CONTROLFILE;
```

Recovering an Offline Tablespace in an Open Database After the tablespaces of interest are taken offline, you can start open–database, offline–tablespace recovery using the RECOVER command with the TABLESPACE parameter. You can recover one or more offline tablespaces. The remainder of the database may be left open and online for normal database operation.

The following statement recovers two offline tablespaces:

```
RECOVER TABLESPACE ts1, ts2;
```

After the tablespaces that contain the damaged files have been taken offline, and you are positive the associated datafiles are also offline (check the file's status in V$DATAFILE), recover selected datafiles using the RECOVER command with the DATAFILE parameter:

```
RECOVER DATAFILE 'filename1', 'filename2';
```

The SQL command equivalent of Server Manager media recovery options is the SQL command ALTER DATABASE command with the RECOVER clause. Generally, database recovery should be performed using Server Manager; which prompts you for information and returns messages from the system. However, if you want to design your own recovery application using SQL commands, use the ALTER DATABASE command.

Starting Recovery During Instance Startup You can start complete media recovery using the STARTUP command with the RECOVER option in Server Manager. After an instance is started, and the database is mounted, complete media recovery proceeds as described in "Complete Media Recovery" on page 24 – 17.

See Also: For information about taking tablespaces offline, see "Taking Tablespaces Offline" on page 8 – 8.

For more information about the STARTUP command, see page 3 – 2.

Applying Redo Log Files

During complete or incomplete media recovery, redo log files (online and archived) are applied to the datafiles during the roll forward phase of media recovery. Because rollback data is recorded in the redo log, rolling forward regenerates the corresponding rollback segments. Rolling forward proceeds through as many redo log files as necessary to bring the database forward in time. As a log file is needed, Oracle suggests the name of the file. For example, if you are using Server Manager, it returns the following lines and prompt:

```
ORA–00279: Change #### generated at DD/MM/YY HH:MM:SS needed for
           thread #
ORA–00289: Suggestion : logfile
ORA–00280: Change #### for thread # is in sequence #
Specify log: [<RET> for suggested | AUTO | FROM logsource |
   CANCEL ]
```

Similar messages are returned when using an ALTER DATABASE... RECOVER statement. However, no prompt is displayed.

Applying Log Files

This section describes how log files can be applied in different environments.

Suggested Log Filenames Oracle suggests log filenames by concatenating the current values of the initialization parameters

LOG_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT and using information from the control file. Therefore, if all the required archived log files are mounted at LOG_ARCHIVE_DEST, and the value for LOG_ARCHIVE_FORMAT is never altered, Oracle can suggest and apply log files to complete media recovery automatically without your intervention. If the location specified by LOG_ARCHIVE_DEST is not available (for example, because of media failure), you can change the value for this parameter, move the log files to the new location, and start a new instance before beginning media recovery.

In some cases, you might want to override the current setting for LOG_ARCHIVE_DEST as a source for log files. For example, assume that a database is open and an offline tablespace must be recovered, but not enough space is available to mount the necessary log files at the location specified by LOG_ARCHIVE_DEST. In this case, you can mount the log files to an alternate location, then specify the alternate location to Oracle for the recovery operation. To specify the location where required log files can be found, use the LOGSOURCE parameter of the SET command in Server Manager. Use the RECOVER...FROM parameter of the ALTER DATABASE command in SQL.

> **Note:** Overriding the log source does not affect the archive log destination for filled online groups being archived.

Consider overriding the current setting for LOG_ARCHIVE_DEST when not enough space is available to mount all the required log files at any one location. In this case, you can set the log file source to an operating system variable (such as a logical or an environment variable) that acts as a search path to several locations.

See Also: Such functionality is operating system–dependent. See your operating system–specific Oracle documentation for more information.

Applying Log Files when Using Server Manager If the suggested archived redo log file is correct, apply the suggested archived redo log. You do not have to specify a filename unless the suggested file is incorrect. After a filename is provided, Oracle applies the redo log file to roll forward the restored datafiles.

In Server Manager, you can have Oracle automatically apply the redo log files that it suggests by choosing either of the following options:

- Before starting media recovery, issue the following Server Manager statement to turn on automatic recovery:

  ```
  SET AUTORECOVERY ON;
  ```

  Automatic application of the suggested redo log starts once recovery begins.

- After media recovery is started, enter “auto” when prompted for a redo log file. Automatic application of the suggested redo log starts from this point.

Suggested redo log files are automatically applied until a suggested redo log is incorrect or recovery is complete. You might need to specify online redo log files manually when using cancel–based recovery or a backup of the control file.

See Also: For examples of logfile application, see your operating system–specific Oracle documentation.

Application of Log Files When Using SQL Commands Application of redo log files is similar to the application of log files. However, a prompt for log files is not displayed after media recovery is started. Instead, you must provide the correct log file using an ALTER DATABASE RECOVER LOGFILE statement. For example, if a message suggests LOG1.ARC, you can apply the suggestion using the following statement:

```
ALTER DATABASE RECOVER LOGFILE ’log1.arc’;
```

As a result, recovering a tablespace requires several statements, as indicated in the following example (DBA input is boldfaced; variable information is italicized.):

```
> ALTER DATABASE RECOVER TABLESPACE users;
ORA–00279: Change #### generated at DD/MM/YY HH:MM:SS needed for
           thread #
ORA–00289: Suggestion : logfile1
ORA–00280: Change #### for thread # is in sequence #
> ALTER DATABASE RECOVER LOGFILE ’logfile1’;
ORA–00279: Change #### generated at DD/MM/YY HH:MM:SS needed for
           thread #<D%0>
ORA–00289: Suggestion : logfile2
ORA–00280: Change #### for thread # is in sequence #
```

```
> ALTER DATABASE RECOVER LOGFILE ’logfile2’;
(Repeat until all logs are applied.)
Statement processed.
> ALTER TABLESPACE users ONLINE;
Statement processed.
```

In this example, it is assumed that the backup files have been restored, and that the user has administrator privileges.

Like the method you used with Server Manager, automatic application of the redo logs can be started with the following statements, before and during recovery, respectively:

```
ALTER DATABASE RECOVER AUTOMATIC ...;

ALTER DATABASE RECOVER AUTOMATIC LOGFILE suggested_log_file;
```

An example of the first statement follows:

```
> ALTER DATABASE RECOVER AUTOMATIC TABLESPACE users;
Statement processed.
> ALTER TABLESPACE users ONLINE;
Statement processed.
```

In this example, it is assumed that the backup files have been restored, and that the user has administrator privileges.

An example of the ALTER DATABASE RECOVER AUTOMATIC LOGFILE statement follows:

```
> ALTER DATABASE RECOVER TABLESPACE users;
ORA–00279: Change #### generated at DD/MM/YY HH:MM:SS needed for
           thread #
ORA–00289: Suggestion : logfile1
ORA–00280: Change #### for thread # is in sequence #
> ALTER DATABASE RECOVER AUTOMATIC LOGFILE ’logfile1’;
Statement processed.
> ALTER TABLESPACE users ONLINE;
Statement processed.
```

In this example, assume that the backup files have been restored, and that the user has administrator privileges.

> **Note:** After issuing the ALTER DATABASE RECOVER command, you can view all files that have been considered for recovery in the V$RECOVERY_FILE_STATUS view. You can access status information for each file in the V$RECOVERY_STATUS view. These views are not accessible after you terminate the recovery session.

See Also: For information about the content of all recovery–related views, see the *Oracle7 Server Reference.*

Successful Application of Redo Logs If you are using Server Manager's recovery options (not SQL statements), each time Oracle finishes applying a redo log file, the following message is returned:

```
Log applied.
```

Make sure that the message "Log applied" is returned after each application of a redo log file. If the suggested file is incorrect or you provide an incorrect filename, an error message is returned instead. If you see an error message instead of "Log applied," a redo log file required for recovery has not been applied. Recovery cannot continue until the required redo log file is applied.

If an error message is returned after supplying a redo log filename, one of the following errors has been detected:

- If the error message says that the file cannot be found, you may have entered the wrong filename. Re–enter the correct filename.
- If the redo log file is found, but cannot be opened, then it may be locked. After unlocking the redo log file, re–enter the filename.
- If a redo log file is found and opened, but cannot be read, an I/O error is returned. In this case, the redo log file may have been only partially written or may have been corrupted. If you can locate an uncorrupted or complete copy of the log, you can simply apply that copy; you do not need to restart recovery. Otherwise, if no other copy of the log exists and you know the time of the last valid redo entry, you can perform time–based or change–based recovery; in this case, you must restart recovery from the beginning, including restoring backups.

Interrupting Media Recovery

If you start a media recovery operation and must then interrupt it (for example, because a recovery operation must end for the night and resume the next morning), you can interrupt recovery at any time by taking either of the following actions:

- Enter the word "cancel" when prompted for a redo log file.
- If you must abort when recovering an individual datafile, or when automated recovery is in progress, use your operating system's interrupt signal.

After recovery is canceled, it must be completed before opening a database for normal operation. To resume recovery, restart it. Recovery resumes where it left off when it was canceled.

⚠ **Warning:** There are several reasons why, after starting recovery, you may want to restart. If, for example, you want to restart with a different backup or want to use the same backup, but need to change the end–time to an earlier point–in–time than you initially specified, then *the entire operation must recommence by restoring a backup*. Failure to do so may result in "file inconsistent" error messages when attempting to open the database.

Restoring a Full Backup, NOARCHIVELOG Mode

If a database is in NOARCHIVELOG mode and a media failure damages some or all of the datafiles, usually the only option for recovering the database is to restore the most recent full backup. If you are using Export to supplement regular backups, you can instead restore the database by importing an exported backup of the database.

The disadvantage of NOARCHIVELOG mode is that to recover your database from the time of the most recent full backup up to the time of the media failure, you have to re–enter manually all of the changes executed in that interval. However, if your database was in ARCHIVELOG mode, the redo log covering this interval would have been available as archived log files or online log files. This would have enabled you to use complete or incomplete recovery to reconstruct your database and minimize the amount of lost work.

If you have a database damaged by media failure and operating in NOARCHIVELOG mode, and you want to restore from your most recent full backup (your only option at this point), perform the following tasks.

To Restore the Most Recent Full Backup (NOARCHIVELOG Mode)

1. If the database is open, shut it down using the Server Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. If the hardware problem that caused the media failure has been corrected so that the backup database files can be restored to their original locations, follow only Step 2.1 before proceeding to Step 3. If, on the other hand, the hardware problem has not been corrected and some or all of the database files must be restored to alternative locations, follow Steps 2.1 through 2.4.

   2.1 Restore the most recent full backup. All of the datafiles and control files of the full backup must be restored, not just the damaged files. This guarantees that the entire database is synchronized to a single point in time.

   2.2 If necessary, edit the restored parameter file to indicate the new location of the control files.

2.3 Start an instance using the restored and edited parameter file and mount, but do not open, the database.

2.4 Perform the steps necessary to record the relocation of the restored datafiles as described in "Renaming and Relocating Datafiles" on page 9 – 7. If applicable, perform the steps necessary to record the relocation of online redo log files, as described in "Renaming and Relocating Online Redo Log Members" on page 5 – 6.

3. Issue the ALTER DATABASE OPEN RESETLOGS command, which opens the database and resets the current log sequence to 1. It also invalidates all redo entries in the online redo log file. Restoring from a full backup and then resetting the log discards all changes to the database made from the time the backup was taken to the time of the media failure.

See Also: See "Using the Export and Import Utilities for Supplemental Database Protection" on page 23 – 18.

Specifying Parallel Recovery

The RECOVERY_PARALLELISM initialization parameter specifies the number of concurrent recovery processes to use for any recovery operation. Because crash recovery occurs at instance startup, this parameter is useful for specifying the number of processes to use for crash recovery. The value of this parameter is also the default number of processes used for media recovery if the PARALLEL clause of the RECOVER command is not specified. The value of this parameter must be greater than one and cannot exceed the value of the PARALLEL_MAX_SERVERS parameter.

In general, parallel recovery is most effective at reducing recovery time when several datafiles on several different disks are being recovered concurrently. Crash recovery (recovery after instance failure) and media recovery of many datafiles on different disk drives are good candidates for parallel recovery. Parallel recovery requires a minimum of eight recovery processes to improve upon serial recovery.

See Also: For more information on parallel recovery, see *Oracle7 Server Concepts*.

For more information about initialization parameters, see the *Oracle7 Server Reference*.

Performing Complete Media Recovery

This section describes the steps necessary to complete media recovery operations, and includes the following topics:

- Performing Closed Database Recovery
- Performing Open–Database, Offline–Tablespace Recovery
- Performing Open–Database, Offline–Tablespace Individual Recovery

Do not depend solely on the steps in the following procedures to understand all the tasks necessary to recover from a media failure. If you haven't already done so, familiarize yourself with the fundamental recovery concepts and strategies on page 24 – 2.

See Also: See page 24 – 47 for a detailed list of the different problems that media failures can cause and describes the appropriate methods of recovery from each type of problem.

Performing Closed Database Recovery

This section describes steps to perform closed database recovery of either all damaged datafiles in one operation, or individual recovery of each damaged datafile in separate operations.

To Perform Closed Database Recovery

1. If the database is open, shut it down using the Server Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.

2. If you're recovering from a media error, correct it if possible.

   ☞ **Attention:** If the hardware problem that caused the media failure was temporary, and the data was undamaged (for example, a disk or controller power failure), *stop at this point*.

3. If files are permanently damaged, restore the most recent backup files (taken as part of a full or partial backup) of *only* the datafiles damaged by the media failure. Do not restore any undamaged datafiles or any online redo log files. If the hardware problem has been repaired, and damaged datafiles can be restored to their original locations, do so, and skip Step 6 of this procedure. If the hardware problem persists, restore the datafiles to an alternative storage device of the database server and continue with this procedure.

   Note: If you do not have a backup of a specific datafile, you might be able to create an empty replacement file that can be recovered.

4. Start Server Manager and connect to Oracle with administrator privileges.

5. Start a new instance and mount, but do not open, the database using either the Server Manager Startup Database dialog box (with the Startup Mount radio button selected), or the STARTUP command with the MOUNT option.

6. If one or more damaged datafiles were restored to alternative locations in Step 3, the new location of these files must be indicated to the control file of the associated database. Therefore, use the operation described in "Renaming and Relocating Datafiles" on page 9 – 7, as necessary.

7. All datafiles you want to recover must be online during complete media recovery. To get the datafile names, check the list of datafiles that normally accompanies the current control file, or query the V$DATAFILE view. Then, issue the ALTER DATABASE command with the DATAFILE ONLINE option to ensure that all datafiles of the database are online. For example, to guarantee that a datafile named USERS1 (a fully specified filename) is online, enter the following statement:

   ```
   ALTER DATABASE DATAFILE 'users1' ONLINE;
   ```

   If a specified datafile is already online, Oracle ignores the statement.

8. To start closed database recovery of all damaged datafiles in one step, use either the Server Manager Apply Recovery Archive dialog box, or an equivalent RECOVER DATABASE statement.

   8.1 To start closed database recovery of an individual damaged datafile, use the RECOVER DATAFILE statement in Server Manager.

   Note: For maximum performance, use parallel recovery to recover the datafiles.

9. Now Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle prompts you for each required redo log file.

Oracle continues until all required archived redo log files have been applied to the restored datafiles. The online redo log files are then automatically applied to the restored datafiles and notifies you when media recovery is complete. If no archived redo log files are required for complete media recovery, Oracle does not prompt for any. Instead, all necessary online redo log files are applied, and media recovery is complete.

After performing closed database recovery, the database is recovered up to the moment that media failure occurred. You can then open the database using the SQL command ALTER DATABASE with the OPEN option.

See Also: See "Restoring Damaged Datafiles" on page 24 – 8 for more information about creating datafiles.

For more information about datafile lists, see "Listing Database Files Before Backup" on page 23 – 8.

For more information about applying redo log files, see "Applying Redo Log Files" on page 24 – 11.

Performing Open–Database, Offline–Tablespace Recovery

At this point, an open database has experienced a media failure, and the database remains open while the undamaged datafiles remain online and available for use. The damaged datafiles are automatically taken offline by Oracle.

This procedure cannot be used to perform complete media recovery on the datafiles of the SYSTEM tablespace. If the media failure damages any datafiles of the SYSTEM tablespace, Oracle automatically shuts down the database.

See Also: To proceed with complete media recovery, follow the procedure in "Performing Closed Database Recovery" on page 24 – 18.

To Perform Open–Database, Offline–Tablespace Recovery

1. The starting point for this recovery operation can vary, depending on whether you left the database open after the media failure occurred.

   1.1 *If the database was shut down*, start a new instance, and mount and open the database. Perform this operation using the Server Manager Startup Database dialog box (with the Startup Open radio button selected), or with the STARTUP command with the OPEN option. After the database is open, take all tablespaces that contain damaged datafiles offline.

   1.2 *If the database is still open* and only damaged datafiles of the database are offline, take all tablespaces containing damaged datafiles offline. Oracle identifies damaged datafiles via error messages. Tablespaces can be taken offline using either the Take Offline menu item of Server Manager, or the SQL command ALTER TABLESPACE with the OFFLINE option, as described in

"Taking Tablespaces Offline" on page 8 – 8. If possible, take the damaged tablespaces offline with temporary priority (to minimize the amount of recovery).

2. Correct the hardware problem that caused the media failure. If the hardware problem cannot be repaired quickly, you can proceed with database recovery by restoring damaged files to an alternative storage device.

3. If files are permanently damaged, restore the most recent backup files (taken as part of a full or partial backup) of *only* the datafiles damaged by the media failure. Do not restore undamaged datafiles, online redo log files, or control files. If the hardware problem has been repaired and the datafiles can be restored to their original locations, do so. If the hardware problem persists, restore the datafiles to an alternative storage device of the database server.

    Note: If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

4. If one or more damaged datafiles were restored to alternative locations (Step 3), indicate the new locations of these files to the control file of the associated database by using the procedure in "Renaming and Relocating Datafiles" on page 9 – 7, as necessary.

5. After connecting with administrator privileges, use the RECOVER TABLESPACE statement in Server Manager to start offline tablespace recovery of all damaged datafiles in one or more offline tablespaces using one step.

    Note: For maximum performance, use parallel recovery to recover the datafiles.

6. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the applying of files is automated, Oracle prompts for each required redo log file.

    Oracle continues until all required archived redo log files have been applied to the restored datafiles. The online redo log files are then automatically applied to the restored datafiles to complete media recovery.

    If no archived redo log files are required for complete media recovery, Oracle does not prompt for any. Instead, all necessary online redo log files are applied, and media recovery is complete.

7. The damaged tablespaces of the open database are now recovered up to the moment that media failure occurred. You can bring the offline tablespaces online using the Place Online menu item of Server Manager, or the SQL command ALTER TABLESPACE with the ONLINE option.

See Also: For more information about redo log application, see "Applying Redo Log Files" on page 24 – 11.

For more information about creating datafiles, see "Restoring Damaged Datafiles" on page 24 – 8.

Performing Open–Database, Offline–Tablespace Individual Recovery

Identical to the preceding operation, here an open database has experienced a media failure, and remains open while the undamaged datafiles remain online and available for use. The damaged datafiles are automatically taken offline by Oracle.

> **Note:** This procedure cannot be used to perform complete media recovery on the datafiles of the SYSTEM tablespace. If the media failure damages any datafiles of the SYSTEM tablespace, Oracle automatically shuts down the database.

To Perform Open–Database, Offline–Tablespace Individual Recovery

1. The starting point for this recovery operation can vary, depending on whether you left the database open after the media failure occurred.

   1.1 *If the database was shut down*, start a new instance, and mount and open the database. Perform this operation using the Server Manager Startup Database dialog box (with the Startup Open radio button selected), or with the STARTUP command with the OPEN option. After the database is open, take all tablespaces that contain damaged datafiles offline.

   1.2 *If the database is still open* and only damaged datafiles of the database are offline, take all tablespaces containing damaged datafiles offline. Oracle identifies damaged datafiles via error messages. Tablespaces can be taken offline using either the Take Offline menu item of Server Manager, or the SQL command ALTER TABLESPACE with the OFFLINE option, as described in "Taking Tablespaces Offline" on page 8 – 8. If possible, take the damaged tablespaces offline with temporary priority (to minimize the amount of recovery).

2. Correct the hardware problem that caused the media failure. If the hardware problem cannot be repaired quickly, you can proceed with database recovery by restoring damaged files to an alternative storage device.

3. If files are permanently damaged, restore the most recent backup files (taken as part of a full or partial backup) of *only* the datafiles damaged by the media failure. Do not restore undamaged datafiles, online redo log files, or control files. If the hardware problem has been repaired and the datafiles can be restored to their original locations, do so. If the hardware problem persists, restore the datafiles to an alternative storage device of the database server.

   Note: If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

4. If one or more damaged datafiles were restored to alternative locations (Step 3), indicate the new locations of these files to the control file of the associated database by using the procedure in "Renaming and Relocating Datafiles" on page 9 – 7, as necessary.

5. After connecting with administrator privileges use the RECOVER DATAFILE statement in Server Manager to start recovery of an individual damaged datafile in an offline tablespace

   Note: For maximum performance, use parallel recovery to recover the datafiles.

6. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle prompts for each required redo log file.

   Oracle continues until all required archived redo log files have been applied to the restored datafiles. The online redo log files are then automatically applied to the restored datafiles to complete media recovery.

   If no archived redo log files are required for complete media recovery, Oracle does not prompt for any. Instead, all necessary online redo log files are applied, and media recovery is complete.

7. The damaged tablespaces of the open database are now recovered up to the moment that media failure occurred. You can bring the offline tablespaces online using the Place Online menu item of Server Manager, or the SQL command ALTER TABLESPACE with the ONLINE option.

See Also: For information about how to proceed with complete media recovery, see “Performing Closed Database Recovery” on page 24 – 18.

For more information about creating datafiles, see “Restoring Damaged Datafiles” on page 24 – 8.

Performing Incomplete Media Recovery

This section descrines the steps necessary to complete the different types of incomplete media recovery operations, and includes the following topics:

- Performing Cancel–Based Recovery
- Performing Time–Based Recovery
- Performing Change–Based Recovery

See Also: Do not rely solely on this section to understand the procedures necessary to recover from a media failure. Also see “Examples of Media Failures and Appropriate Recovery Procedures” on page 24 – 47 for a detailed list of the different types of problems that media failures can cause, and the appropriate methods of recovery from each type of problem.

Changing the System Time on a Running Database

If your database is affected by seasonal time changes (for example, daylight savings time), you may experience a problem if a time appears twice in the redo log and you want to recover to the second, or later time. To deal with time changes, perform cancel–based or change–based recovery to the point in time where the clock is set back, then continue with the time–based recovery to the exact time.

Performing Cancel–Based Recovery

This section describes how to perform cancel–based recovery.:

To Perform Cancel–Based Recovery

1. If the database is still open and incomplete media recovery is necessary, shut down the database using the Server Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. Make a full backup of the database (all datafiles, a control file, and the parameter files of the database) as a precautionary measure, in case an error is made during the recovery procedure.
3. If a media failure occurred, correct the hardware problem that caused the media failure.

4. If the current control files do not match the physical structure of the database at the intended time of recovery (for example, if a datafile was added after the point in time to which you intend to recover), then restore a backup of the control file that reflects the database's physical file structure (contains the names of datafiles and online redo log files) at the point at which incomplete media recovery is intended to finish. Review the list of files that correspond to the current control file as well as each control file backup to determine the correct control file to use. If necessary, replace all current control files of the database with the correct control file backup. You can, alternatively, create a new control file to replace the missing one.

   > **Note**: If a database control file cannot function or be replaced with a control file backup, you must edit the parameter file associated with the database to modify the CONTROL_FILES parameter.

5. Restore backup files (taken as part of a full or partial backup) of *all* the datafiles of the database. All backup files used to replace existing datafiles must have been taken before the intended time of recovery. For example, if you intend to recover to redo log sequence number 38, then restore all datafiles with backups completed before redo log sequence number 38.

   If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

   If a datafile was added after the intended time of recovery, it is not necessary to restore a backup for this file, as it will no longer be used for the database after recovery is complete.

   If the hardware problem that caused a media failure has been solved and all datafiles can be restored to their original locations, do so, and skip Step 8 of this procedure. If a hardware problem persists, restore damaged datafiles to an alternative storage device.

   > **Note:** Files in read–only tablespaces should be offline if you are using a control file backup. Otherwise, recovery will try to update the headers of the read–only files.

6. Start Server Manager and connect to Oracle with administrator privileges.

7. Start a new instance and mount the database. You can perform this operation using the Server Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.

8. If one or more damaged datafiles were restored to alternative locations in Step 5, the new locations of these files must be indicated to the control file of the associated database.
9. If a backup of the control file is being used with this incomplete recovery (that is, a control file backup or re–created control file was restored in Step 4), indicate this in the dialog box or command used to start recovery (that is, specify the USING BACKUP CONTROLFILE parameter).
10. Use Server Manager Apply Recovery Archives dialog box, or an equivalent RECOVER DATABASE UNTIL CANCEL statement to begin cancel–based recovery.
11. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle supplies the name it expects to find from LOG_ARCHIVE_DEST and requests you to stop or proceed with applying the log file. If the control file is a backup file, you must supply names of online logs.

    Oracle continues to apply redo log files.
12. Continue applying redo log files until the most recent, undamaged redo log file has been applied to the restored datafiles.
13. Enter "CANCEL" to cancel recovery after Oracle has applied the redo log file just prior to the damaged file. Cancel–based recovery is now complete.

    Oracle returns a message indicating whether recovery is successful.

Opening the Database After Successful Cancel–Based Recovery

The first time you open the database subsequent to incomplete media recovery, you must explicitly specify whether to reset the log sequence number by including either the RESETLOGS or NORESETLOGS option. Resetting the redo log:

- discards any redo information that was not applied during recovery, ensuring that it will never be applied
- reinitializes the control file information about online redo logs and redo threads
- clears the contents of the online redo logs
- creates the online redo log files if they do not currently exist
- resets the log sequence number to 1

⚠ **Warning:** Resetting the redo log discards all changes to the database made since the first discarded redo information. Updates entered after that time must be re–entered manually.

Use the following rules when deciding to specify RESETLOGS or NORESETLOGS:

- Reset the log sequence number if you used a backup of the control file in recovery, no matter what type of recovery was performed (complete or incomplete).
- Reset the log sequence number if the recovery was actually incomplete. For example, you must have specified a previous time or SCN, not one in the future.
- Do not reset logs if recovery was complete (unless you used a backup control file). This applies when you intentionally performed complete recovery and when you performed incomplete recovery but actually recovered all changes in the redo logs anyway. See the explanation in step 12 for how to examine the ALERT file to see if incomplete recovery was actually complete.
- Do not reset logs if you are using the archived logs of this database for a standby database. If the log must be reset, then you will have to re–create your standby database.

To preserve the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN NORESETLOGS option. To reset the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN RESETLOGS option. (If you attempt to reset the log when you should not, or if you neglect to reset the log when you should, Oracle returns an error and does not open the database. Correct the error and try again.)

If the log sequence number is reset when opening a database, different messages are returned, depending on whether the recovery was complete or incomplete. If the recovery was complete, the following message appears in the ALERT file:

```
RESETLOGS after complete recovery through change scn
```

If the recovery was incomplete, the following message is reported in the ALERT file:

```
RESETLOGS after incomplete recovery UNTIL CHANGE scn
```

If you reset the redo log sequence when opening the database, immediately shut down the database normally and make a full database backup. Otherwise, you will not be able to recover changes made after you reset the logs. Until you take a full backup, the only way to recover will be to repeat the procedures you just finished, up to resetting the logs. (You do not need to back up the database if you did not reset the log sequence.)

After opening the database using the RESETLOGS option, check the ALERT log to see if Oracle7 has detected inconsistencies between the data dictionary and the control file (for example, a datafile that the data dictionary includes but does not list in the new control file).

If a datafile exists in the data dictionary but not in the new control file, Oracle7 creates a placeholder entry in the control file under MISSING*nnnn* (where *nnnn* is the file number in decimal). MISSING*nnnn* is flagged in the control file as being offline and requiring media recovery. The actual datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn*, so that it points to the datafile only when the datafile was read–only or offline normal. If, on the other hand, MISSING*nnnn* corresponds to a datafile that was not read–only or offline normal, then the rename operation cannot be used to make the datafile accessible, because the datafile requires media recovery that is precluded by the results of RESETLOGS. In this case, the tablespace containing the datafile must be dropped.

In contrast, if a datafile indicated in the control file is not present in the data dictionary, Oracle7 removes references to it from the new control file. In both cases, Oracle7 includes an explanatory message in the ALERT file to let you know what was found.

See Also: See "Creating Additional Copies of the Control File, and Renaming and Relocating Control Files" on page 6 – 4.

For more information about creating datafiles, see "Restoring Damaged Datafiles" on page 24 – 8.

To relocate or rename datafiles, see "Renaming and Relocating Datafiles" on page 9 – 7, as necessary.

For more information about listing datafiles, see "Listing Database Files Before Backup" on page 23 – 8.

For more information about applying redo logs, see "Applying Redo Log Files" on page 24 – 11.

Performing Time–Based Recovery

When you are performing time–based, incomplete media recovery, and you are recovering with a backup control file and have read–only tablespaces, contact Oracle Support before attempting this recovery procedure.

To Perfrom Time–Based Recovery

1. If the database is still open and incomplete media recovery is necessary, shut down the database using the Server Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.

2. Make a full backup of the database (all datafiles, a control file, and the parameter files of the database) as a precautionary measure, in case an error is made during the recovery procedure.

3. If a media failure occurred, correct the hardware problem that caused the media failure.

4. If the current control files do not match the physical structure of the database at the intended time of recovery (for example, if a datafile was added after the point in time to which you intend to recover), then restore a backup of the control file that reflects the database's physical file structure (contains the names of datafiles and online redo log files) at the point at which incomplete media recovery is intended to finish. Review the list of files that corresponds to the current control file and each control file backup to determine the correct control file to use. If necessary, replace all current control files of the database with the correct control file backup. You can, alternatively, create a new control file to replace the missing one.

   > **Note**: If a database control file cannot function or be replaced with a control file backup because the hardware problem causing the media failure persists, you must edit the parameter file associated with the database to modify the CONTROL_FILES parameter.

5. Restore backup files (taken as part of a full or partial backup) of *all* the datafiles of the database. All backup files used to replace existing datafiles must have been taken before the intended time of recovery. For example, if you intend to recover to redo log sequence number 38, then restore all datafiles with backups completed before redo log sequence number 38.

   If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

   If a datafile was added after the intended time of recovery, it is not necessary to restore a backup for this file, as it will no longer be used for the database after recovery is complete.

   If the hardware problem that caused a media failure has been solved and all datafiles can be restored to their original locations, do so, and skip Step 8 of this procedure. If a hardware problem persists, restore damaged datafiles to an alternative storage device.

> **Note:** Files in read–only tablespaces should be offline if you are using a control file backup. Otherwise, the recovery will try to update the headers of the read–only files.

6. Start Server Manager and connect to Oracle with administrator privileges.
7. Start a new instance and mount the database. This operation can be performed with the Server Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.
8. If one or more damaged datafiles were restored to alternative locations in Step 5, the new locations of these files must be indicated to the control file of the associated database.
9. All datafiles of the database must be online unless an offline tablespace was taken offline normally. To get the names of all datafiles to recover, check the list of datafiles that normally accompanies the control file being used or query the V$DATAFILE view. Then, use the ALTER DATABASE command and the DATAFILE ONLINE option to make sure that all datafiles of the database are online. For example, to guarantee that a datafile named USERS1 (a fully specified filename) is online, enter the following statement:

   ```
   ALTER DATABASE DATAFILE ’users1’ ONLINE;
   ```

   If a backup of the control file is being used with this incomplete recovery (that is, a control file backup or re–created control file was restored), indicate this in the dialog box or command used to start recovery. If a specified datafile is already online, Oracle ignores the statement.
10. Issue the RECOVER DATABASE UNTIL TIME statement to begin time–based recovery. The time is always specified using the following format, delimited by single quotation marks: ’YYYY–MM–DD:HH24:MI:SS’.
11. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle supplies the name it expects to find from LOG_ARCHIVE_DEST and requests you to stop or proceed with applying the log file. If the control file is a backup file, you must supply names of online logs. Oracle continues to apply redo log files.

12. Continue applying redo log files until the last required redo log file has been applied to the restored datafiles. Oracle automatically terminates the recovery when it reaches the correct time, and returns a message indicating whether recovery is successful.

Opening the Database After Successful Time–Based Recovery

The first time you open the database subsequent to incomplete media recovery, you must explicitly specify whether to reset the log sequence number by including either the RESETLOGS or NORESETLOGS option. Resetting the redo log:

- discards any redo information that was not applied during recovery, ensuring that it will never be applied
- reinitializes the control file information about online redo logs and redo threads
- clears the contents of the online redo logs
- creates the online redo log files if they do not currently exist
- resets the log sequence number to 1

⚠ **Warning:** Resetting the redo log discards all changes to the database made since the first discarded redo information. Updates entered after that time must be re–entered manually.

Use the following rules when deciding to specify RESETLOGS or NORESETLOGS:

- Reset the log sequence number if you used a backup of the control file in recovery, no matter what type of recovery was performed (complete or incomplete).
- Reset the log sequence number if the recovery was actually incomplete. For example, you must have specified a previous time or SCN, not one in the future.
- Do not reset logs if recovery was complete (unless you used a backup control file). This applies when you intentionally performed complete recovery and when you performed incomplete recovery but actually recovered all changes in the redo logs anyway. See the explanation in step 12 for how to examine the ALERT file to see if incomplete recovery was actually complete.
- Do not reset logs if you are using the archived logs of this database for a standby database. If the log must be reset, then you will have to re–create your standby database.

To preserve the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN NORESETLOGS option. To reset the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN RESETLOGS option. (If you attempt to reset the log when you should not, or if you neglect to reset the log when you should, Oracle returns an error and does not open the database. Correct the error and try again.)

If the log sequence number is reset when opening a database, different messages are returned, depending on whether the recovery was complete or incomplete. If the recovery was complete, the following message appears in the ALERT file:

```
RESETLOGS after complete recovery through change scn
```

If the recovery was incomplete, the following message is reported in the ALERT file:

```
RESETLOGS after incomplete recovery UNTIL CHANGE scn
```

If you reset the redo log sequence when opening the database, immediately shut down the database normally and make a full database backup. Otherwise, you will not be able to recover changes made after you reset the logs. Until you take a full backup, the only way to recover will be to repeat the procedures you just finished, up to resetting the logs. (You do not need to back up the database if you did not reset the log sequence.)

After opening the database using the RESETLOGS option, check the ALERT log to see if Oracle7 has detected inconsistencies between the data dictionary and the control file (for example, a datafile that the data dictionary includes but does not list in the new control file).

If a datafile exists in the data dictionary but not in the new control file, Oracle7 creates a placeholder entry in the control file under MISSING*nnnn* (where *nnnn* is the file number in decimal). MISSING*nnnn* is flagged in the control file as being offline and requiring media recovery. The actual datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn*, so that it points to the datafile only when the datafile was read–only or offline normal. If, on the other hand, MISSING*nnnn* corresponds to a datafile that was not read–only or offline normal, then the rename operation cannot be used to make the datafile accessible, because the datafile requires media recovery that is precluded by the results of RESETLOGS. In this case, the tablespace containing the datafile must be dropped.

In contrast, if a datafile indicated in the control file is not present in the data dictionary, Oracle7 removes references to it from the new control file. In both cases, Oracle7 includes an explanatory message in the ALERT file to let you know what was found.

See Also: See "Creating Additional Copies of the Control File, and Renaming and Relocating Control Files" on page 6 – 4.

For more information about creating datafiles, see "Restoring Damaged Datafiles" on page 24 – 8.

To relocate or rename datafiles, see "Renaming and Relocating Datafiles" on page 9 – 7, as necessary.

For more information about listing datafiles, see "Listing Database Files Before Backup" on page 23 – 8.

For more information about applying redo logs, see "Applying Redo Log Files" on page 24 – 11.

Performing Change–Based Recovery

This section describes how to perform change–based recovery.

To Perform Change–Based Recovery

1. If the database is still open and incomplete media recovery is necessary, shut down the database using the Server Manager Shutdown Abort mode of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. Make a full backup of the database (all datafiles, a control file, and the parameter files of the database) as a precautionary measure, in case an error is made during the recovery procedure.
3. If a media failure occurred, correct the hardware problem that caused the media failure.
4. If the current control files do not match the physical structure of the database at the intended time of recovery (for example, if a datafile was added after the point in time to which you intend to recover), then restore a backup of the control file that reflects the database's physical file structure (contains the names of datafiles and online redo log files) at the point at which incomplete media recovery is intended to finish. Review the list of files that correspond to the current control file as well as each control file backup to determine the correct control file to use. If necessary, replace all current control files of the database with the correct control file backup. You can, alternatively, create a new control file to replace the missing one.

    Note: If a database control file cannot function or be replaced with a control file backup, you must edit the parameter file associated with the database to modify the CONTROL_FILES parameter.

5. Restore backup files (taken as part of a full or partial backup) of *all* the datafiles of the database. All backup files used to replace existing datafiles must have been taken before the intended time of recovery. For example, if you intend to recover to redo log sequence number 38, then restore all datafiles with backups completed before redo log sequence number 38.

   If you do not have a backup of a specific datafile, you can create an empty replacement file, which can be recovered.

   If a datafile was added after the intended time of recovery, it is not necessary to restore a backup for this file, as it will no longer be used for the database after recovery is complete.

   If the hardware problem that caused a media failure has been solved and all datafiles can be restored to their original locations, do so, and skip Step 8 of this procedure. If a hardware problem persists, restore damaged datafiles to an alternative storage device.

   > **Note:** Files in read–only tablespaces should be offline if you are using a control file backup. Otherwise, recovery will try to update the headers of the read–only files.

6. Start Server Manager and connect to Oracle with administrator privileges.

7. Start a new instance and mount the database. You can perform this operation using the Server Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.

8. If one or more damaged datafiles were restored to alternative locations in Step 5, the new locations of these files must be indicated to the control file of the associated database.

9. To get the names of all datafiles to recover, check the list of datafiles that normally accompany the control file being used or query the V$DATAFILE view. Then, use the ALTER DATABASE command with the DATAFILE ONLINE option to make sure that all datafiles of the database are online. For example, to guarantee that a datafile named USERS1 (a fully specified filename) is online, enter the following statement:

   ```
   ALTER DATABASE DATAFILE ’users1’ ONLINE;
   ```

   If a specified datafile is already online, Oracle ignores the statement.

If a backup of the control file is being used with this incomplete recovery (that is, a control file backup or re–created control file was restored), specify the USING BACKUP CONTROLFILE parameter in the dialog box or command used to start recovery.

10. Issue the RECOVER DATABASE UNTIL CHANGE statement to begin change–based recovery. The SCN is specified as a decimal number without quotation marks.
11. Oracle begins the roll forward phase of media recovery by applying the necessary redo log files (archived and online) to reconstruct the restored datafiles. Unless the application of files is automated, Oracle supplies the name it expects to find from LOG_ARCHIVE_DEST and requests you to stop or proceed with applying the log file. If the control file is a backup file, you must supply names of online logs. Oracle continues to apply redo log files.
12. Continue applying redo log files until the last required redo log file has been applied to the restored datafiles. Oracle automatically terminates the recovery when it reaches the correct time, and returns a message indicating whether recovery is successful.

Opening the Database After Successful Change–Based Recovery

The first time you open the database subsequent to incomplete media recovery, you must explicitly specify whether to reset the log sequence number by including either the RESETLOGS or NORESETLOGS option. Resetting the redo log:

- discards any redo information that was not applied during recovery, ensuring that it will never be applied
- reinitializes the control file information about online redo logs and redo threads
- clears the contents of the online redo logs
- creates the online redo log files if they do not currently exist
- resets the log sequence number to 1

⚠ **Warning:** Resetting the redo log discards all changes to the database made since the first discarded redo information. Updates entered after that time must be re–entered manually.

Use the following rules when deciding to specify RESETLOGS or NORESETLOGS:

- Reset the log sequence number if you used a backup of the control file in recovery, no matter what type of recovery was performed (complete or incomplete).
- Reset the log sequence number if the recovery was actually incomplete. For example, you must have specified a previous time or SCN, not one in the future.
- Do not reset logs if recovery was complete (unless you used a backup control file). This applies when you intentionally performed complete recovery and when you performed incomplete recovery but actually recovered all changes in the redo logs anyway. See the explanation in step 12 for how to examine the ALERT file to see if incomplete recovery was actually complete.
- Do not reset logs if you are using the archived logs of this database for a standby database. If the log must be reset, then you will have to re–create your standby database.

To preserve the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN NORESETLOGS option. To reset the log sequence number when opening a database after recovery, use the SQL command ALTER DATABASE with the OPEN RESETLOGS option. (If you attempt to reset the log when you should not, or if you neglect to reset the log when you should, Oracle returns an error and does not open the database. Correct the error and try again.)

If the log sequence number is reset when opening a database, different messages are returned, depending on whether the recovery was complete or incomplete. If the recovery was complete, the following message appears in the ALERT file:

```
RESETLOGS after complete recovery through change scn
```

If the recovery was incomplete, the following message is reported in the ALERT file:

```
RESETLOGS after incomplete recovery UNTIL CHANGE scn
```

If you reset the redo log sequence when opening the database, immediately shut down the database normally and make a full database backup. Otherwise, you will not be able to recover changes made after you reset the logs. Until you take a full backup, the only way to recover will be to repeat the procedures you just finished, up to resetting the logs. (You do not need to back up the database if you did not reset the log sequence.)

After opening the database using the RESETLOGS option, check the ALERT log to see if Oracle7 has detected inconsistencies between the data dictionary and the control file (for example, a datafile that the data dictionary includes but does not list in the new control file).

If a datafile exists in the data dictionary but not in the new control file, Oracle7 creates a placeholder entry in the control file under MISSING*nnnn* (where *nnnn* is the file number in decimal). MISSING*nnnn* is flagged in the control file as being offline and requiring media recovery. The actual datafile corresponding to MISSING*nnnn* can be made accessible by renaming MISSING*nnnn*, so that it points to the datafile only when the datafile was read–only or offline normal. If, on the other hand, MISSING*nnnn* corresponds to a datafile that was not read–only or offline normal, then the rename operation cannot be used to make the datafile accessible, because the datafile requires media recovery that is precluded by the results of RESETLOGS. In this case, the tablespace containing the datafile must be dropped.

In contrast, if a datafile indicated in the control file is not present in the data dictionary, Oracle7 removes references to it from the new control file. In both cases, Oracle7 includes an explanatory message in the ALERT file to let you know what was found.

See Also: See "Creating Additional Copies of the Control File, and Renaming and Relocating Control Files" on page 6 – 4.

For more information about creating datafiles, see "Restoring Damaged Datafiles" on page 24 – 8.

To relocate or rename datafiles, see "Renaming and Relocating Datafiles" on page 9 – 7, as necessary.

For more information about listing datafiles, see "Listing Database Files Before Backup" on page 23 – 8.

For more information about applying redo logs, see "Applying Redo Log Files" on page 24 – 11.

Preparing for Disaster Recovery

This section describes how to plan for and implement disaster recovery procedures for your primary database, and includes the following topics:

- Planning and Creating a Standby Database
- Altering the Physical Structure of the Primary Database

Planning and Creating a Standby Database

A *standby database* maintains a duplicate, or standby copy of your primary (also known as *production*) database and provides continued primary database availability in the event of a disaster (when all media is destroyed at your production site). A standby database is constantly in recovery mode. If a disaster occurs, you can take the standby database out of recovery mode and activate it for online use. A standby database is intended *only* for recovery of the primary database; you cannot query or open it for any purpose other than to activate disaster recovery. Once you activate your standby database, you cannot return it to standby recovery mode unless you re–create it as another standby database.

⚠ **Warning:** Activating a standby database resets the online logs of the standby database. Hence, after activation, the logs from your standby database and production database are incompatible.

You must place the data files, log files, and control files of your primary and standby databases on separate physical media. Therefore, it is impossible to use the same control file for both your primary and standby databases.

Creating a Standby Database

This section lists the steps and rules to follow when creating a standby database.

To Create a Standby Database

1. Back up (either online or offline) the data files from your primary database.
2. Create the control file for your standby database by issuing the ALTER DATABASE CREATE STANDBY CONTROLFILE AS 'filename' command, which creates a modified copy of the primary database's control file.
3. Archive the current online logs of the primary database by issuing the ALTER SYSTEM ARCHIVE LOG CURRENT command. Issuing the ALTER SYSTEM ARCHIVE LOG

CURRENT command also ensures consistency among the data files in step 1, the control file in step 2, and the log files.

4. Transfer the standby database control file, archived log files, and backed up data files to the remote (standby) site using operating system commands or utilities. Use an appropriate method if transferring binary files.

⚠ **Warning:** Oracle encourages you to use a datafile naming scheme that keeps the datafile names the same at both the primary and standby databases. If this is not possible, then you can use the datafile name conversion parameters. If you do not use either of these suggested datafile naming schemes, you may end up crashing your standby database.

See Also: For information about setting name conversion parameters when you create your standby database, see "Converting Data File and Log File Names."

Maintaining a Standby Database

This section provides the tasks for maintaining your standby database, including information about clearing standby logfiles.

To Maintain Your Standby Database in Recovery Mode

1. Start up the Oracle instance at the standby database using the NO MOUNT clause.
2. Issue the ALTER DATABASE MOUNT STANDBY DATABASE [EXCLUSIVE / PARALLEL] command.
3. Transfer the archived redo logs from the primary database to the remote (standby) site. Use an appropriate operating system utility for transferring binary data.
4. Place the standby database in recovery mode by issuing the RECOVER [FROM 'location'] STANDBY DATABASE command.

Note: As the archived logs are generated, you must continually transfer and apply them to the standby database. Also, you can only apply logs that have been archived at the primary database to the standby database.

Clearing Online Logfiles You can clear standby database online logfiles to optimize performance as you maintain your standby database. If you prefer not to perform this operation during maintenance, the online logfiles will be cleared automatically during activation. You can clear logfiles using the following statement:

```
ALTER DATABASE CLEAR LOGFILE GROUP integer;
```

Converting Data File and Log File Names

You can set the following initialization parameters so that all filenames from your primary database control file are converted for use by your standby database:

- DB_FILE_STANDBY_NAME_CONVERT
- LOG_FILE_STANDBY_NAME_CONVERT

If your primary and standby databases exist on the same machine (of course, they should not, but if they are), setting these parameters is advisable, because they allow you to make your standby database filenames distinguishable from your primary database filenames.

The DB_FILE_STANDBY_NAME_CONVERT and LOG_FILE_STANDBY_NAME_CONVERT parameters must have two strings. The first string is a sequence of characters to be looked for in a primary database filename. If that sequence of characters is matched, it is replaced by the second string to construct the standby database filename.

Figure 24 – 1 shows how the filename conversion parameters work:

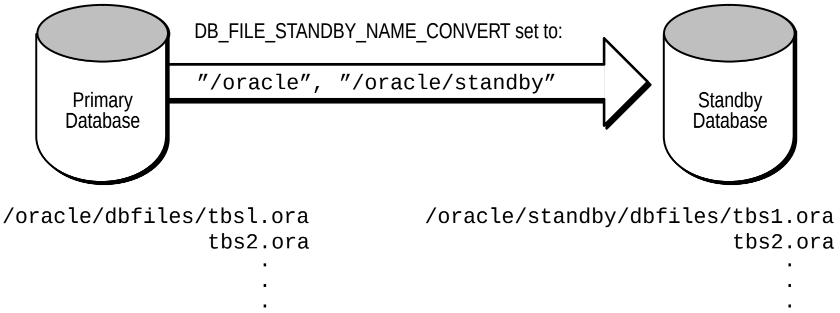

DB_FILE_STANDBY_NAME_CONVERT set to:
"/oracle", "/oracle/standby"
Primary Database
Standby Database
/oracle/dbfiles/tbsl.ora
tbs2.ora
/oracle/standby/dbfiles/tbs1.ora
tbs2.ora

Figure 24 – 1 Setting Filename Conversion Parameters

> **Note:** If you perform a data file (or log file) RENAME at the standby database, or use the AS clause with the ALTER DATABASE CREATE FILE command, then the conversion parameters will not apply to that file.

Activating a Standby Database

In the event of a disaster, you should (if possible) archive your primary database logs (ALTER SYSTEM ARCHIVE LOG CURRENT), transfer them to your standby site, and apply them before activating your standby database. This makes your standby database current to the same point in time as your primary database (before the failure). If you cannot archive your current online logs, then you must activate the standby database without recovering the transactions from the unarchived logs of the primary database.

After you activate your standby database, its online redo logs are reset. Note that this makes the logs from the standby database and primary database incompatible. Also, the standby database is dismounted when activated, therefore, you are unable to look at tables and views immediately after activation.

To Activate a Standby Database

1. Ensure that your standby database is mounted in EXCLUSIVE mode.
2. Issue the ALTER DATABASE ACTIVATE STANDBY DATABASE command.
3. Shut down your standby instances.
4. As soon as possible, back up your new production database. At this point, the former standby database is now your production database. This task, while not required, is a recommended safety measure, because you cannot recover changes made after activation without a backup.
5. Startup the new production instance.

> **Note:** After you activate your standby database, all transactions from unarchived logs at your original production database are lost.

Altering the Physical Structure of the Primary Database

Altering the physical structure of your primary database can have an impact on your standby database. The following sections describe the effects of primary database structural alterations on a standby database.

Adding Data Files

Adding a data file to your primary database generates redo information that, when applied at your standby database, automatically adds the data file name to the standby control file. If the standby database locates the new file with the new filename, the recovery process continues. If the standby database is unable to locate the new data file, the recovery process will stop.

If the recovery process stops, then perform *either* of the following procedures before resuming the standby database recovery process:

- Copy a backup of the added data file from the primary database to the standby database.
- Issue the ALTER DATABASE CREATE DATAFILE command at the standby database.

If you don't want the new data file in the standby database, you can take it offline using the DROP option.

See Also: For more information on offline data file alterations, see "Taking Data Files in the Standby Database Offline" on page 24 – 44.

Renaming Files

Data file renames on your primary database do not take effect at the standby database until the standby database control file is refreshed. If you want the data files at your primary and standby databases to remain in sync when you rename primary database data files, then perform analogous operations on the standby database.

Altering Log Files

You can add log file groups or members to the primary database without affecting your standby database. Likewise, you can drop log file groups or members from the primary database without affecting your standby database. Similarly, enabling and disabling of threads at the primary database has no effect on the standby database.

You may want to keep the online log file configuration the same at the primary and standby databases. If so, when you enable a log file thread with the ALTER DATABASE ENABLE THREAD at the primary database, you should create a new control file for your standby database before activating it. See "Refreshing the Standby Database Control File" on page 24 – 45 for refresh procedures.

If you clear log files at the primary database by issuing the ALTER DATABASE CLEAR UNARCHIVED LOGFILE command, or open the primary database using the RESETLOGS option, you invalidate the standby database. Because the standby database recovery process will not have the archived logs it requires to continue, you will need to re–create the standby database.

Altering Control Files

If you use the CREATE CONTROLFILE command at the primary database to perform any of the following, you may invalidate the standby database's control file:

- change the maximum number of redo log file groups or members
- change the maximum number of data files

- change the maximum number of instances that can concurrently mount and open the database

If you've invalidated the standby database's control file, you must re–create it using the procedures in "Refreshing the Standby Database Control File" on page 24 – 45.

Using the CREATE CONTROLFILE command with the RESETLOGS option on your primary database will force the next open of the primary database to reset the online logs, thereby invalidating the standby database.

Configuring Initialization Parameters

Most initialization parameters at your primary and standby databases should be identical. Specific initialization parameters such as CONTROL_FILES and DB_FILE_STANDBY_NAME_CONVERT should be changed. Differences in other initialization parameters may cause performance degradation at the standby database, and in some cases, bring standby database operations to a halt.

The following initialization parameters play a key role in the standby database recovery process:

- COMPATIBLE

  The COMPATIBLE parameter must be the same at the primary and standby databases. If it is not, you may not be able to apply the logs from your primary database to your standby database.

- DB_FILES

  MAXDATAFILES must be the same at both databases so that you allow the same number of files at the standby database as you allow at the primary database.

- CONTROL_FILES

  CONTROL_FILES must be different between the primary and standby databases. The names of the control files that you list in this parameter for the standby database must exist at the standby database.

- DB_FILE_STANDBY_NAME_CONVERT (or LOG_FILE_STANDBY_NAME_CONVERT)

  Set the DB_FILE_STANDBY_NAME_CONVERT (or LOG_FILE_STANDBY_NAME_CONVERT) parameter when you want to make your standby database filenames distinguishable from your primary database filenames. For more information on this parameter see "Converting Data File and Log File Names" on page 24 – 40.

See Also: For more information on initialization parameters, see the *Oracle7 Server Reference.*

Taking Data Files in the Standby Database Offline

You can take standby database datafiles offline as a means to support a subset of your primary database's datafiles. For example, you decide it is undesirable to recover the primary database's temporary tablespaces on the standby database. So you take the datafiles offline using the ALTER DATABASE DATAFILE 'fn' OFFLINE DROP command on the standby database. If you do this, then the tablespace containing the offline files must be dropped after opening the standby database.

Performing Direct Path Operations

When you perform a direct load originating from either direct path load, table create via subquery, or index create on the primary database, the performance improvement applies *only* to the primary database; there is no corresponding recovery process performance improvement on the standby database. The standby database recovery process still sequentially reads and applies the redo information generated by the unrecoverable direct load.

Primary database processes using the UNRECOVERABLE option are not propagated to the standby database. Why? Because these processes do not appear in the archived redo logs. If you want to propagate such processes to your standby database, perform any *one* of the following tasks.

To Propagate UNRECOVERABLE Processes to a Standby Database

1. Take the affected datafiles offline in the standby database, and drop the tablespace after activation.
2. Re–create the standby database from a new database backup.
3. Back up the affected tablespace and archive the current logs in the primary database. Transfer the datafiles to the standby database. Then resume standby recovery. This is the same procedure that you would perform to guarantee ordinary database recoverability after an UNRECOVERABLE operation.

If you perform an unrecoverable operation at the primary database, and attempt to recover at the standby database, you will not receive error messages during recovery. Such error messages appear in the standby database alert log. Thus, you should check the standby database alert log periodically.

See Also: For more details, see “Taking Datafiles in the Standby Database Offline” on page 24 – 44.

Refreshing the Standby Database Control File

The following steps describe how to refresh, or create a copy of changes you’ve made to the primary database control file.

To Refresh the Standby Database Control File

1. Issue the CANCEL command on the standby database to halt its recovery process.
2. Shut down the standby instances.
3. Issue the ALTER DATABASE CREATE STANDBY CONTROLFILE AS ’filename’ statement on the primary database to create the control file for the standby database.
4. Issue the ALTER SYSTEM ARCHIVE LOG CURRENT statement on the primary database to archive the current online logs of your primary database.
5. Transfer the standby control file and archived log files to the standby site.
6. Restart and mount (but do not open) the standby database by issuing the ALTER DATABASE MOUNT STANDBY DATABASE [EXCLUSIVE / PARALLEL] statement.
7. Restart the recovery process on the standby database by issuing the RECOVER [FROM ’location’] STANDBY DATABASE statement.

Unrecoverable Objects and Recovery

You can create tables and indexes using the CREATE TABLE AS SELECT command. You can also specify that Oracle create them as *unrecoverable*. When you create a table or index as unrecoverable, Oracle does not generate redo log records for the operation. Thus, objects created unrecoverable cannot be recovered, even if you are running in ARCHIVELOG mode.

> **Note:** If you cannot afford to lose tables or indexes created unrecoverable, take a backup after the unrecoverable table or index is created.

Be aware that when you perform a media recovery, and some tables or indexes are created as recoverable while others are unrecoverable, the unrecoverable objects will be marked logically corrupt by the RECOVER operation. Any attempt to access the unrecoverable objects returns an ORA–01578 error message. You should drop the unrecoverable objects, and recreate them, if needed.

Because it is possible to create a table unrecoverable and then create a recoverable index on that table, the index is not marked as logically corrupt after you perform a media recovery. However, the table was unrecoverable (and thus marked as corrupt after recovery), so the index points to corrupt blocks. The index must be dropped, and the table and index must be re–created if necessary.

See Also: For information about the impact of UNRECOVERABLE operations on a standby database, see page 24 – 44.

Read–Only Tablespaces and Recovery

This section describes how read–only tablespaces affect instance and media recovery.

Using a Backup Control File

Media recovery with the USING BACKUP CONTROLFILE option checks for read–only files. It is an error to attempt recovery of a read–only file. You can avoid this error by taking all datafiles from read–only tablespaces offline before doing recovery with a backup control file. Therefore, it is very important to have the correct version of the control file for the recovery. If the tablespace will be read–only when the recovery is complete, then the control file must be from a time when the tablespace was read–only. Similarly, if the tablespace will be read–write at the end of recovery, it should be read–write in the control file. If the appropriate control file is not available, you should create a new control file with the CREATE CONTROLFILE command.

Re–Creating a Control File

If you need to re–create a control file for a database with read–only tablespaces, you must follow some special procedures. Issue the ALTER DATABASE BACKUP CONTROLFILE TO TRACE command to get a listing of the procedure that you need to follow. The procedure is similar to the procedure for offline normal tablespaces, except that you need to bring the tablespace online after the database is open.

Re–creating a control file can also affect the recovery of read–write tablespaces that were at one time read–only. If you re–create the control file after making the tablespace writeable, Oracle can no longer determine when the tablespace was changed from read–only to read–write. Thus, you can no longer recover from the read–only version of the tablespace. Instead, you must recover from the time of the most recent backup. It is important to backup a tablespace immediately after making it read–write.

Examples of Recovery Procedures

This section describes how to recover from common media failures, and includes the following topics:

- Types of Media Failures
- Loss of Datafiles
- Loss of Online Redo Log Files
- Loss of Archived Redo Log Files
- Loss of Control Files
- Recovery From User Errors

Types of Media Failures

Media failures fall into two general categories: permanent and temporary. Permanent media failures are serious hardware problems that cause the permanent loss of data on the disk. Lost data cannot be recovered except by repairing or replacing the failed storage device and restoring backups of the files stored on the damaged storage device. Temporary media failures are hardware problems that make data temporarily inaccessible; they do not corrupt the data. Following are two examples of temporary media failures:

- A disk controller fails. Once the disk controller is replaced, the data on the disk can be accessed.

- Power to a storage device is cut off. Once the power is returned, the storage device and all associated data is accessible again.

Loss of Datafiles

If a media failure affects datafiles of a database, the appropriate recovery procedure depends on the archiving mode of the database, the type of media failure, and the exact files affected by the media failure. The following sections explain the appropriate recovery strategies in various situations.

Loss of Datafiles, NOARCHIVELOG Mode

If either a permanent or temporary media failure affects *any* datafiles of a database operating in NOARCHIVELOG mode, Oracle automatically shuts down the database. Depending on the type of media failure, you can use one of two recovery paths:

- If the media failure is temporary, correct the temporary hardware problem and restart the database. Usually, instance recovery is possible, and all committed transactions can be recovered using the online redo log.
- If the media failure is permanent, follow the steps on page 24 – 15 to recover from the media failure.

Loss of Datafiles, ARCHIVELOG Mode

If either a permanent or temporary media failure affects the datafiles of a database operating in ARCHIVELOG mode, the following situations can exist:

- If a temporary or permanent media failure affects any datafiles of the SYSTEM tablespace or any datafiles that contain active rollback segments, the database becomes inoperable and should be immediately shut down if it has not already been shut down by Oracle.

  If the hardware problem is temporary, correct the problem and restart the database. Usually, instance recovery is possible, and all committed transactions can be recovered using the online redo log.

  If the hardware problem is permanent, follow the procedure in "Performing Closed Database Recovery" on page 24 – 18.

- If a temporary or permanent media failure affects only datafiles not mentioned in the previous item, the affected datafiles are unavailable and taken offline automatically by Oracle, but the database can continue to operate.

  If the unaffected portions of the database must remain available, do not shut down the database. First take all tablespaces that contain problem datafiles offline using the temporary option.

Then follow the procedure in “Performing Open Database–Offline Tablespace Recovery” on page 24 – 20.

Loss of Online Redo Log Files

If a media failure has affected the online redo log of a database, the appropriate recovery procedure depends on the configuration of the online redo log (mirrored or non–mirrored), the type of media failure (temporary or permanent), and the types of online redo log files affected by the media failure (current, active, not yet archived, or inactive online redo log files). The following sections describe the appropriate recovery strategies in various situations.

Loss of an Online Redo Log Member of Mirrored Online Redo Log

If the online redo log of a database is mirrored, and at least one member of each online redo log group is not affected by the media failure, Oracle allows the database to continue functioning as normal (error messages are written to the LGWR trace file and ALERT file of the database). However, you should handle the problem by taking one of the following actions:

- If the hardware problem is temporary, correct the problem. After it has been fixed, LGWR accesses the previously unavailable online redo log files as if the problem never existed.
- If the hardware problem is permanent, use the DROP command to drop the damaged member and use the ADD command to add a new member.

  Note: The newly added member provides no redundancy until the log group is reused.

Loss of All Online Redo Log Members of an Online Redo Log Group

If all members of an online redo log group are damaged by a media failure, different situations can arise, depending on the type of online redo log group affected by the failure and the archiving mode of the database. You can locate the filename in V$LOGFILE, and then look for the group number corresponding to the one you lost to verify the lost file’s status (verify that it was inactive).

```
SELECT *
FROM v$logfile
;

GROUP#     STATUS          MEMBER
------------------------------------------------
0001                        log1
0002                        log2
0003                        log3

SELECT *
FROM v$log
```

```
;

GROUP#  MEMBERS            STATUS      ARCHIVED
-------------------------------------------------
 0001     1                  INACTIVE     YES
 0002     1                  ACTIVE       YES
 0003     1                  CURRENT      NO
```

Loss of an Inactive, Online Redo Log Group If all members of an inactive online redo log group are damaged, the following situations can arise:

- If a temporary media failure affects only an inactive online redo log group, correct the problem; LGWR can reuse the group when required.
- If a media failure permanently prevents access to only an inactive online redo log group, the damaged inactive online redo log group will eventually halt normal database operation.

  If you notice the problem before the database shuts down, use the ALTER DATABASE CLEAR LOGFILE command.

  If the database has already shut down, perform the following tasks:

To Recover From Loss of an Inactive, Online Redo Log Group

1. Abort the current instance immediately with the Server Manager Shutdown Database dialog box with the Shutdown Abort radio button selected, or the SHUTDOWN command with the ABORT option.
2. Start a new instance and mount the database, but do not open it. This operation can be performed with the Server Manager Startup Database dialog box with the Startup Mount radio button selected, or the STARTUP command with the MOUNT option.
3. If the lost log was archived, issue the ALTER DATABASE CLEAR LOGFILE command.
4. If the lost log was unarchived, issue the ALTER DATABASE CLEAR UNARCHIVED LOGFILE command, and immediately backup the database. Also backup the database's control file (using the ALTER DATABASE command with the BACKUP CONTROLFILE option).

   Clearing a log that has not been archived allows it to be reused without archiving it. However, this will make backups unusable if they were started before the last change in the log (unless the file was taken offline prior to the first change in the log). Hence, if the cleared logfile is needed for recovery of a backup, it will not be possible to recover that backup.

If there is an offline datafile that requires the cleared unarchived log to bring it online, the keywords UNRECOVERABLE DATAFILE are required. The datafile and its entire tablespace will have to be dropped from the database because the redo necessary to bring it online is being cleared, and there is no copy of it.

Note: The ALTER DATABASE CLEAR LOGFILE command could fail (with an I/O error due to media failure) in two cases:

- When it is not possible to relocate the logfile onto alternative media by re–creating it under the currently configured logfile name.
- When it is not possible to reuse the currently configured logfile name to recreate the logfile because the name itself is invalid or unusable (for example, due to media failure).

In these two cases, the CLEAR LOGFILE command (before receiving the I/O error) would have successfully updated the control file to change the state of the logfile to "being cleared" and "not requiring archiving." The I/O error occurred at the step in which CLEAR LOGFILE attempts to create the new logfile and write zeros to it.

At this point, you can complete recovery by executing, in order, the following commands:

- ADD a logfile under a new name.
- DROP the logfile under the old name.

You can now open the database.

Loss of an Active Online Redo Log Group If your database is still running and the lost active log is not the current log, you can use the ALTER SYSTEM CHECKPOINT command. If successful, your active log is rendered inactive, and you can follow the steps on page 24 – 49.

If unsuccessful, or if your database has already halted, you cannot use the steps on page 24 – 49. Instead, perform the following tasks:

To Recover From Loss of an Active Online Redo Log Group

1. If the media failure is temporary, correct the problem so that Oracle can reuse the group when required.
2. If the database is in NOARCHIVELOG mode and a permanent media failure prevents access to an active online redo log group, restore the database from a full backup.

   After restoring the database, redo the work and open the database using the RESETLOGS option. Updates done after the backup have been lost and must be re–executed. Shut down the database and take a full offline backup.
3. If the database was in ARCHIVELOG mode, incomplete media recovery must be performed. Use the procedure given in "Performing Cancel–Based, Time–Based, or Change–Based Recovery" on page 24 – 24, recovering up through the log before the damaged log. Ensure that the current name of the lost redo log can be used for a newly created file. If not, issue the RENAME command to rename the damaged online redo log group to a new location.
4. Open the database using the RESETLOGS option.

Note: All updates executed from the endpoint of the incomplete recovery to the present must be re–executed.

Loss of Multiple Redo Log Groups If you have lost multiple groups of the online redo log, use the recovery method for the most difficult log to recover. The order of difficulty, from most difficult to least, follows:

1. the current online redo log
2. the active online redo log
3. the unarchived redo log
4. the inactive online redo log

Loss of Archived Redo Log Files

If the database is operating so that filled online redo log groups are being archived, and the only copy of an archived redo log file is damaged, it does not affect the present operation of the database. However, the following situations can arise if media recovery is required in the future:

- If *all* datafiles have been backed up after the filled online redo log group (which is now archived) was written, the archived version of the filled online redo log group is not required for complete media recovery operation.

- Assume the most recent backup file of a datafile was taken before the filled online redo log group was written. The group now corresponds to the damaged archived redo log file. At some future point, the corresponding datafile is damaged by a permanent media failure. The most recent backup of the damaged datafile must be used, and incomplete media recovery can only recover the database up to the damaged archived redo log file.
- If time–based recovery is needed, the damaged archived redo log file may be required if you use old datafile backups that were taken before the original online redo log group was written. In this case, the incomplete media recovery can only recover the database up to the damaged archived redo log group.

⚠ **Warning:** If you know that an archived redo log group has been damaged, immediately backup all datafiles so that you will have a complete backup that does not require the damaged archived redo log.

Loss of Control Files

If a media failure has affected the control files of a database (whether control files are mirrored or not), the database continues to run until the first time that an Oracle background process needs to access the control files. At this point, the database and instance are automatically shut down.

If the media failure is temporary and the database has not yet shut down, immediately correcting the media failure can avoid the automatic shut down of the database. However, if the database shuts down before the temporary media failure is corrected, you can restart the database after fixing the problem (and restoring access to the control files).

The appropriate recovery procedure for media failures that permanently prevent access to control files of a database depends on whether you have mirrored the control files. The following sections describe the appropriate procedures.

Loss of a Member of a Mirrored Control File

Use the following steps to recover a database after one or more control files of a database have been damaged by a permanent media failure, and at least one control file has not been damaged by the media failure:

Note: If all control files of a mirrored control file configuration have been damaged, follow the instructions for recovering from the loss of non–mirrored control files.

To Recover a Database After Control Files Are Damaged

1. If the instance is still running, immediately abort the current instance with the Server Manager Shutdown Abort option of the Shutdown Database dialog box, or the SHUTDOWN command with the ABORT option.
2. Correct the hardware problem that caused the media failure. If the hardware problem cannot be repaired quickly, you can proceed with database recovery by restoring damaged control files to an alternative storage device.
3. Use an intact copy of the database's control file to copy over the damaged control files. If possible, copy the intact control file to the original locations of all damaged control files. If the hardware problem persists, copy the intact control file to alternative locations. If you restored *all* damaged control files to their original location, proceed to Step 5. If all damaged control files were not restored, or not restored to their original location, proceed to Step 4.
4. If all damaged control files were not restored, or not restored to their original location in Step 3, the parameter file of the database must be edited so that the CONTROL_FILES parameter reflects the current locations of all control files and excludes all control files that were not restored.
5. Start a new instance. Mount and open the database.

Loss of All Copies of the Current Control File

If all control files of a database have been lost or damaged by a permanent media failure, but all online redo logfiles remain intact, you can recover by creating a new control file (using the CREATE CONTROLFILE command with the NORESETLOGS option). Then execute RECOVER DATABASE followed by ALTER DATABASE OPEN.

Depending on the existence and currency of a control file backup, you have the following options for generating the text of the CREATE CONTROLFILE command:

- If you have executed ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS since you made the last structural change to the database, and if you have saved the SQL command output, then you can use the CREATE CONTROLFILE command from the output as–is. If, however, your most recent execution of ALTER DATABASE BACKUP CONTROLFILE TO TRACE was performed before you made a structural change to the database, then you must edit the output of ALTER DATABASE BACKUP CONTROLFILE TO TRACE to reflect that structural change. For example, if you recently added a datafile to the database, then you should add that datafile to the DATAFILE clause of the CREATE CONTROLFILE command.

- If you have not backed up the control file using the TO TRACE option, but instead have used the TO *filename* option of the ALTER DATABASE BACKUP CONTROLFILE command, then you can use the control file copy to obtain SQL command output. You can do this by copying the backup control file and executing STARTUP MOUNT before executing ALTER DATABASE BACKUP CONTROLFILE TO TRACE NORESETLOGS. If your control file copy predated a recent structural change, you must edit the TO TRACE output to reflect that structural change.
- If you do not have a backup of the control file (in either TO TRACE format or TO *filename* format), then you must generate the CREATE CONTROLFILE command manually.

Recovery From User Errors

An accidental or erroneous operational or programmatic change to the database can cause loss or corruption of data. Recovery may require a return to a state prior to the error.

> **Note:** If the database administrator has properly granted powerful privileges (such as DROP ANY TABLE) to only selected, appropriate users, user errors that require database recovery are minimized.

To Recover Data Lost or Corrupted by User Error

1. Back up the existing, intact database.
2. Leave the existing database intact, but reconstruct a temporary copy of the database up to the time of the user error using time–based recovery.
3. Export the lost or corrupted data from the reconstructed, temporary copy of the database.
4. Import the lost or corrupted data into the permanent database.
5. Delete the files associated with the temporary copy of the reconstructed database to conserve disk space.

The following scenario describes how to recover a table that has been accidentally dropped.

1. The database that experienced the user error can remain online and available for normal use. The database can remain open or be shut down. Back up all datafiles of the existing database in case an error is made during the remaining steps of this procedure.
2. Create a temporary copy of the database to a past point–in–time using time–based recovery. Be careful not to cause a conflict with the existing control file of the permanent database. Restore a single control file backup to an alternative location (step 4) and edit the parameter file, as necessary, or create a new control file at the alternative location. Also, restore all datafiles to alternative locations (step 5) so that you do not affect the permanent copy of the database.
3. Export the lost data using the Oracle utility Export from the temporary, restored version of the database. In this case, export the accidentally dropped table.

   Note: System audit options are exported.
4. Import the exported data (step 3) into the permanent copy of the database using the Oracle Import utility.
5. Delete the files of the temporary, reconstructed copy of the database to conserve space.

See Also: For more information about the Import and Export utilities, see *Oracle7 Server Utilities.*

PART VI

Reference

APPENDIX

A

Space Estimations for Schema Objects

This appendix contains equations that can help you approximate the amount of space for specific schema objects. Constants in estimate calculations are operating system–specific.

☞ **Attention:** While these equations help estimate schema object size, they are *approximations*, and may vary from your actual results

Estimating Space Required by Non–Clustered Tables

The procedures in this section describe how to estimate the total number of data blocks necessary to hold data inserted into a non–clustered table Within this sample calculation, no concurrency is assumed, and users are not performing intervening delete or update operations.

> **Note:** This is a best case scenario *only* when users insert rows without performing deletes or updates.

Typically, the space required to store a set of rows will exceed this calculation when updates and deletes are also being performed on the table. The actual space required for complex workloads is best determined empirically, and then scaled by the number of rows in the table. In general, increasing amounts of concurrent activity on the same data block results in additional overhead (for transaction records), so it is important that you take into account such activity when scaling empirical results.

To Calculate Space Required by Non–Clustered Tables

1. Calculate the total block header size.
2. Calculate the available data space per data block.
3. Calculate the space used per row.
4. Calculate the total number of rows that will fit in a data block.

Step 1: Calculate the Total Block Header Size

The space required by the data block header is the result of the following formula:

```
Space after headers (hsize)
=
DB_BLOCK_SIZE – KCBH – UB4 – KTBBH – (INITRANS – 1) * KTBIT – KDBH
```

Where:

DB_BLOCK_ SIZE	is the database block size as viewed in the V$PARAMETER view
KCBH, UB4, KTBBH, KTBIT,KDBH	are constants whose sizes you can obtain by selecting from entries in the V$TYPE_SIZE view
INITRANS	is the initial number of transaction entries allocated to the table

Step 2: Calculate the Available Data Space Per Data Block

The space reserved in each data block for data, as specified by PCTFREE, is calculated as follows:

```
available data space (availspace)
=
CEIL(hsize * (1 – PCTFREE/100)) – KDBT
```

Where:

CEIL rounds a fractional result to the next highest integer

PCTFREE is the percentage of space reserved for updates in the table

KDBT is a constant whose size you can obtain by selecting the entry from the V$TYPE_SIZE view

Note: If you are unable to locate the value of KDBT, use the value of UB4 instead.

Step 3: Calculate the Space Used per Row

Calculating the amount of space used per row is a multi–step task.

First, you must calculate the column size, including byte lengths:

```
Column size including byte length
=
column size + (1, if column size < 250, else 3)
```

Note: You can also determine column size empirically, by selecting `avg(vsize(colname))` for each column in the table.

Then, calculate the row size:

```
Rowsize
=
row header (3 * UB1) + sum of column sizes including length bytes
```

Finally, you can calculate the space used per row:

```
Space used per row (rowspace)
=
MIN(UB1 * 3 + UB4 + SB2, rowsize) + SB2
```

Where:

UB1, UB4, SB2 are constants whose size can be obtained by selecting entries from the V$TYPE_SIZE view

When the space per row exceeds the available space per data block, but is less than the available space per data block without any space reserved for updates (for example, available space with PCTFREE=0), each row will be stored in its own block.

When the space per row exceeds the available space per data block without any space reserved for updates, rows inserted into the table will be chained into 2 or more pieces, hence, this storage overhead will be higher.

Figure A – 1 depicts elements in a table row.

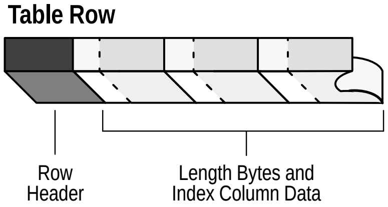


Figure A – 1 Calculating the Size of a Row

Step 4: Calculate the Total Number of Rows That Will Fit in a Data Block

You can calculate the total number of rows that will fit into a data block using the following equation:

```
Number of rows in block
=
FLOOR(availspace / rowspace)
```

Where:

FLOOR rounds a fractional result to the next lowest integer

In summary, remember that this procedure provides a reasonable *estimate* of a table's size, not an exact number of blocks or bytes. After you have estimated the size of a table, you can use this information when specifying the INITIAL storage parameter (size of the table's initial extent) in your corresponding CREATE TABLE statement.

See Also: See your operating system–specific Oracle documentation for any substantial deviations from the constants provided in this procedure.

Space Requirements for Tables in Use

After a table is created and in use, the space required by the table is usually higher than the estimate derived from your calculations. More space is required due to the method by which Oracle manages free space in the database.

Estimating Space for Indexes

The following procedure demonstrates how to estimate the initial amount of space required by an index.

The calculations in the procedure rely on average column lengths of the columns that constitute an index; therefore, if column lengths in each row of a table are relatively constant with respect to the indexed columns, the estimates calculated by the following procedure are more accurate.

To Estimate Space for Indexes

1. Calculate the total block header size.
2. Calculate the available data space per data block.
3. Calculate the combined column lengths of an average index value.
4. Calculate the total average index value size.
5. Calculate the number of blocks and bytes required for the index.

> **Note:** Several calculations are required to obtain a final estimate, and several of the constants (indicated by *****) provided are operating system–specific. Your estimates should not significantly differ from actual values.

See Also: See your operating system–specific Oracle documentation for any substantial deviations from the constants provided in the following procedure.

Step 1: Calculate the Total Block Header Size

Figure A – 2 shows the elements of an index block used in the following calculations. The space required by the data block header of a block to contain index data is given by the formula:

```
block header size = fixed header + variable transaction header
```

where:

fixed header*	113 bytes
variable transaction header*	24**I* *I* is the value of INITRANS for the index.

If INITRANS =2 (the default for indexes), the previous formula can be simplified:

```
block header = 113 + (24*2) bytes
                = 161 bytes
```

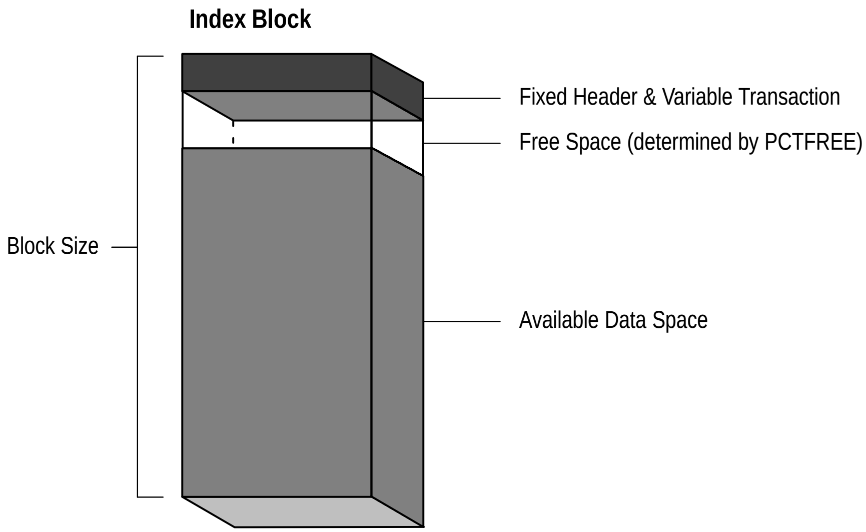

Index Block
Fixed Header & Variable Transaction
Free Space (determined by PCTFREE)
Block Size
Available Data Space

Figure A – 2 Calculating the Space for an Index

Step 2: Calculate Available Data Space Per Data Block

The space reserved in each data block for index data, as specified by PCTFREE, is calculated as a percentage of the block size minus the block header:

```
available
  data            = (block size – block header) –
space per block     ((block size – block header)*(PCTFREE/100))
```

The block size of a database is set during database creation and can be determined using the Server Manager command SHOW, if necessary:

```
SHOW PARAMETERS db_block_size;
```

If the data block size is 2K and PCTFREE=10 for a given index, the total space for new data in data blocks allocated for the index is:

```
available data space per block
         = (2048 bytes – 161 bytes) –
          ((2048 bytes – 161 bytes)*(10/100))
         = (1887 bytes) – (1887 bytes * 0.1)
         = 1887 bytes – 188.7 bytes
         = 1698.3 bytes
```

Step 3: Calculate Combined Column Lengths

The space required by the average value of an index must be calculated before you can complete Step 4, calculating the total row size. This step is identical to Step 3 in the procedure for calculating table size, except you only need to calculate the average combined column lengths of the columns in the index.

Step 4: Calculate Total Average Index Value Size

Figure A – 3 shows elements of an index entry used in the following calculations. Once you have calculated the combined column length of an average index entry, you can calculate the total average entry size according to the following formula:

```
bytes/entry = entry header + ROWID length + F + V + D
```

where:

entry header	2 bytes
ROWID length	6 bytes
F	Total length bytes of all columns that store 127 bytes or less. The number of length bytes required by each column of this type is 1 byte.
V	Total length bytes of all columns that store more than 127 bytes. The number of length bytes required by each column of this type is 2 bytes.
D	Combined data space of all index columns (from Step 3).

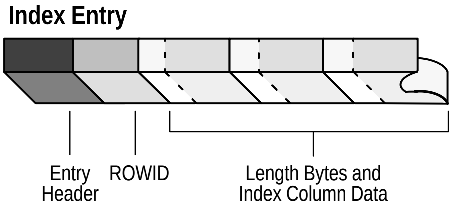


Figure A – 3 Calculating the Average Size of an Index Entry

For example, given that *D* is calculated to be 22 bytes and that the index is comprised of three VARCHAR(10) columns, the total average entry size of the index is:

```
avg. entry size = 2 + 6 + (1 * 3) + (2 * 0) + 22 bytes
                   = 33 bytes
```

Note: For a non–unique index, the ROWID is considered another column, so it must have one length byte.

Step 5: Calculate Number of Blocks and Bytes

Calculate the number of blocks required to store the index using the following formula:

```
# blocks for index =
                              # not null rows
1.05 *  _______________________________________________

        FLOOR(avail. data space per block/avg. entry size)
```

Note: The additional 5% added to this result (by means of the multiplication factor of 1.05) accounts for the extra space required for branch blocks of the index.

For example, continuing with the previous example, and assuming you estimate that indexed table will have 10000 rows that contain non–null values in the columns that constitute the index:

```
# blocks for index =

                           10000 * 33 bytes
1.05 *    _____________________________________

           FLOOR(1700 bytes/33 bytes)*(33 bytes)
```

This results in 204 blocks. The number of bytes can be calculated by multiplying the number of blocks by the data block size.

Remember that this procedure provides a reasonable *estimate* of an index's size, not an exact number of blocks or bytes. Once you have estimated the size of a index, you can use this information when specifying the INITIAL storage parameter (size of the index's initial extent) in your corresponding CREATE INDEX statement.

Temporary Space Required for Index Creation

When creating an index for a loaded table, temporary segments are created to sort the index. The amount of space required to sort an index varies, but can be up to 110% of the size of the index.

Note: Temporary space is not required if the NOSORT option is included in the CREATE INDEX command. However, you cannot specify this option when creating a cluster index.

Estimating Space Required by Clusters

The following procedure demonstrates how to estimate the initial amount of space required by a set of tables in a cluster. This procedure estimates only the initial amount of space required for a cluster. When using these estimates, note that the following items can affect the accuracy of estimations:

- Trailing nulls are not stored, nor is a length byte.
- Inserts of, updates to, and deletes of rows, as well as tables containing columns larger than a single data block can cause fragmentation and chained row pieces. Therefore, the following estimates may tend to be lower that the actual space required if significant fragmentation occurs.

Once you calculate a table's size using the following procedure, you should add about 10 to 20 percent additional space to calculate the initial extent size for a working table.

To Estimate Space Required by Clusters

1. Calculate total block header size and space available for table data.
2. Calculate the combined column lengths of the average rows per cluster key.
3. Calculate the average row size of all clustered tables.
4. Calculate the average cluster block size.
5. Calculate the total number of blocks required for the cluster.

Step 1:
Calculate Total Block Header Size and Space Available for Table Data

The following formula returns the amount of available space in a block:

> **Note:** Several calculations are required to obtain a final estimate, and several of the constants (indicated by *****) provided are operating system–specific. Your estimates should not significantly differ from actual values. See your operating system–specific Oracle documentation for any substantial deviations from the constants provided in the following procedure.

```
space left in block after headers (hspace)
=  BLOCKSIZE – KCBH – UB4 – KTBBH – KTBIT*(INITTRANS – 1) – KDBH
```

where the sizes of KCBH, KTBBH, KTBIT, KDBH, and UB4 can be obtained by selecting * from v$type_size table.

> **Note:** If this is a table segment (instead of the cluster segment shown above), the table directory would simply be 4.

Then use the following formula to calculate the space available for table data:

```
space available for table data
= hspace*(1 – PCTFREE/100) – 4*(NTABLES + 1) * ROWSINBLOCK
```

where:

BLOCKSIZE	is the size of a data block
INITTRANS	is the initial number of transaction entries for the object
PCTFREE	is the percentage of space to reserve in a block for updates
NTABLES	is the number of tables in the cluster
ROWS INBLOCK	is the number of rows in a block

Step 2:
Calculate Space Required by a Row

Use Step 3 from the procedure in "Calculating Space Required by Non–Clustered Tables" to calculate this number. Make note of the following caveats:

- Calculate the data space required by an average row for each table in the cluster. For example, in a cluster that contains tables T1 and T2, calculate the average row size for both tables.
- Do not include the space required by the cluster key in any of the above calculations. However, make note of the space required to store an average cluster key value for Step 5. For example, calculate the data space required by an average row in table T1, not including the space required to store the cluster key.
- Do not include any space required by the row header (that is, the length bytes for each column); this space is accounted for in the next step.

For example, assume two clustered tables are created with the following statements:

```
CREATE TABLE t1 (a CHAR(10), b DATE, c NUMBER(10,2))
   CLUSTER t1_t2 (c);

CREATE TABLE t2 (c NUMBER(10,2), d CHAR(10))
   CLUSTER t1_t2 (c);
```

Notice that the cluster key is column C in each table.

Considering these example tables, the space required for an average row (D_1) of table T1 and the space required for an average row (D_2) of table T2 is:

```
D1 (space/average row)   = (a + b)
                         = (10 + 7) bytes
                         = 17 bytes

D2 (space/average row)   = (d)
                         = 10 bytes
```

Step 3:
Calculate Total Average Row Size

You can calculate the minimum amount of space required by a row in a clustered table according to the following equation:

```
Sn bytes/row = row header + Fn + Vn + Dn
```

where:

row header*	4 bytes per row of a clustered table.
F_n	Total length bytes of all columns in table *n* that store 250 bytes or less. The number of length bytes required by each column of this type is 1 byte.
V_n	Total length bytes of all columns in table *n* that store more than 250 bytes. The number of length bytes required by each column of this type is 3 bytes.
D_n	Combined data space of all columns in table *n* (from Step 3).

Note: Do not include the column length for the cluster key in variables F or V for any table in the cluster. This space is accounted for in Step 5.

For example, the total average row size of the clustered tables T1 and T2 are as follows:

```
S1     = (4 + (1 * 2) + (3 * 0) + 17) bytes
       = 23 bytes

S2     = (4 + (1 * 1) + (3 * 0) + 10) bytes
       = 15 bytes
```

Note: The absolute minimum row size of a clustered row is 10 bytes, and is operating system–specific. Therefore, if your calculated value for a table's total average row size is less than these absolute minimum row sizes, use the minimum value as the average row size in subsequent calculations.

Step 4: Calculate Average Cluster Block Size

To calculate the average cluster block size, first estimate the average number of rows (for all tables) per cluster key. Once this is known, use the following formula to calculate average cluster block size:

$$\text{avg. cluster block size (bytes)=}$$
$$((R_1*S_1) + (R_2*S_2) + .. + (R_n*S_n)) + \text{key header} + C_k + S_k + 2R_t$$

where:

R_n The average number of rows in table *n* associated with a cluster key.

S_n The average row size in table *n* (see Step 4).

key header* 19

C_k Column length for the cluster key.

S_k Space required to store average cluster key value.

R_t Total number of rows associated with an average cluster key (R_1 + R2 ... + R_n). This accounts for the space required in the data block header for each row in the block.

For example, consider the cluster that contains tables T1 and T2. An average cluster key has one row per table T1 and 20 rows per table T2. Also, the cluster key is of datatype NUMBER (column length is 1 byte), and the average number is 4 digits (3 bytes). Considering this information and the previous results, the average cluster key size is:

```
SIZE = ((1 * 23) + (20 * 15) + 19 + 1 + 3 + (2 * 21)) bytes
     = 388 bytes
```

Specify the estimated SIZE in the SIZE option when you create the cluster with the CREATE CLUSTER command. This specifies the space required to hold an average cluster key and its associated rows; Oracle uses the value of SIZE to limit the number of cluster keys that can be assigned to any given data block. After estimating an average cluster key SIZE, choose a SIZE somewhat larger than the average expected size to account for the space required for cluster keys on the high side of the estimate.

To estimate the number of cluster keys that will fit in a database block, use the following formula, which uses the value you calculated in Step 2 for available data space, the number of rows associated with an average cluster key (R_t), and SIZE:

$$\text{\# cluster keys per block}$$
$$= \text{FLOOR(available data space + 2R / SIZE + } 2R_t)$$

For example, with SIZE previously calculated as 400 bytes (calculated as 388 earlier in this step and rounded up), R_t estimated at 21, and available space per data block (from Step 2) calculated as 1742 – 2*R* bytes, the result is as follows:

```
# cluster keys per block
= FLOOR((1936 – 2R + 2R) / (400 + 2 * 21))

= FLOOR(1936 / 442)
= FLOOR(4.4)
= 4
```

Step 5:
Calculate Total Number of Blocks

To calculate the total number of blocks for the cluster, you must estimate the number of cluster keys in the cluster. Once this is estimated, use the following formula to calculate the total number of blocks required for the cluster:

```
# blocks = CEIL(# cluster keys / # cluster keys per block)
```

> **Note:** If you have a test database, you can use statistics generated by the ANALYZE command to determine the number of key values in a cluster key. See “Analyzing Tables, Indexes, and Clusters” on page 16 – 3.

For example, assume that there are approximately 500 cluster keys in the T1_T2 cluster:

```
# blocks T1_T2 = CEIL(500/3)
               = CEIL(166.7)
               = 167
```

To convert the number of blocks to bytes, multiply the number of blocks by the data block size.

This procedure provides a reasonable *estimation* of a cluster’s size, but not an exact number of blocks or bytes. Once you have estimated the space for a cluster, you can use this information when specifying the INITIAL storage parameter (size of the cluster’s initial extent) in your corresponding CREATE CLUSTER statement.

Space Requirements for Clustered Tables in Use

Once clustered tables are created and in use, the space required by the tables is usually higher than the estimate given by the previous section. More space is required due to the method Oracle uses to manage free space in the database.

Estimating Space Required by Hash Clusters

As with index clusters, it is important to estimate the storage required for the data in a hash cluster. Use the procedure described in "Estimating Space Required by Clusters" on page A – 9, with the following additional notes:

- A sub–goal of the procedure is to determine the SIZE of each cluster key. However, for hash clusters, the corresponding sub–goal is to determine the SIZE of each hash key. Therefore, you must consider not only the number of rows per cluster key value, but also the distribution of cluster keys over the hash keys in the cluster.
- In Step 3, make sure to include the space required by the cluster key value. Unlike an index cluster, the cluster key value is stored with each row placed in a hash cluster.
- In Step 5, you are calculating the average hash key size, not cluster key size. Therefore, take into account how many cluster keys map to each hash value. Also, disregard the addition of the space required by the cluster key value, C_k. This value has already been accounted for in Step 3 (see previous item).

Index

A

- abort, shutting down an instance, 3 – 9
- access
  - data
    - managing, 20 – 1
    - system privileges, 20 – 2
  - database
    - controlling, 19 – 1
    - database administrator account, 1 – 4
    - granting privileges, 20 – 12
    - restricting, 3 – 4
    - revoking privileges, 20 – 15
  - object
    - granting privileges, 20 – 13
    - privilege types, 20 – 6
    - revoking privileges, 20 – 15
- accounts
  - operating–system
    - database administrator, 1 – 4
    - role identification, 20 – 20
  - user, SYS and SYSTEM, 1 – 4
- active extents, 17 – 6
- Add Datafiles to Tablespace dialog, 9 – 4
- Add Online Redo Log Group dialog box, 5 – 5
- Add Online Redo Log Member dialog, 5 – 6
- ADMIN OPTION
  - about, 20 – 12
  - revoking, 20 – 15
- AFTER triggers, auditing and, 21 – 23 to 21 – 25
- ALERT file
  - about, 4 – 13
  - location of, 4 – 14
  - session high water mark in, 19 – 6
  - size of, 4 – 14
  - using, 4 – 12
  - when written, 4 – 14
- ALL_INDEXES view, filling with data, 16 – 5
- ALL_TAB_COLUMNS view, filling with data, 16 – 5
- ALL_TABLES view, filling with data, 16 – 5
- allocation
  - extents, 11 – 8
  - extents for clusters, 14 – 9
  - minimizing extents for rollback segments, 17 – 13
  - multi–threaded server and, 4 – 5
  - temporary space, 11 – 5
- alphanumeric datatypes, 10 – 17
- ALTER CLUSTER command
  - ALLOCATE EXTENT option, 14 – 9
  - MAXTRANS option, 10 – 10
  - using for hash clusters, 15 – 8
  - using for index clusters, 14 – 8
- ALTER DATABASE command
  - ADD LOG MEMBER option, 5 – 6
  - ADD LOGFILE option, 5 – 5
  - ARCHIVELOG option, 22 – 5
  - BACKUP CONTROLFILE TO TRACE option, 23 – 15
  - CREATE DATAFILE option, 24 – 8
  - database partially available to users, 3 – 6
  - DATAFILE...OFFLINE DROP option, 9 – 7
  - DROP LOGFILE MEMBER option, 5 – 10
  - DROP LOGFILE option, 5 – 9

MOUNT option, 3 – 6
NOARCHIVELOG option, 22 – 5
NORESETLOGS option, 24 – 27, 24 – 32, 24 – 36
OPEN NORESETLOGS option, 24 – 20
OPEN option, 3 – 7
RENAME FILE option, datafiles for multiple tablespaces, 9 – 9
RESETLOGS option, 24 – 27, 24 – 32, 24 – 36

ALTER FUNCTION command, COMPILE option, 16 – 19

ALTER INDEX command
about, 13 – 8
MAXTRANS option, 10 – 10

ALTER PACKAGE command, COMPILE option, 16 – 19

ALTER PROCEDURE command, COMPILE option, 16 – 19

ALTER PROFILE command
altering resource limits, 19 – 17
COMPOSITE_LIMIT option, 19 – 17

Alter Profile dialog, 19 – 17

ALTER RESOURCE COST command, 19 – 18

ALTER ROLE command, changing authorization method, 20 – 11

Alter Role dialog, 20 – 11

ALTER ROLLBACK SEGMENT command
changing storage parameters, 17 – 9
OFFLINE option, 17 – 12
ONLINE option, 17 – 11, 17 – 12
PUBLIC option, 17 – 10
STORAGE clause, 17 – 9

Alter Rollback Segment Storage dialog, 17 – 9

ALTER SEQUENCE command, 12 – 11

ALTER SESSION command, SET SQL_TRACE parameter, 4 – 12

ALTER SYSTEM command
ARCHIVE LOG ALL option, 22 – 8
ARCHIVE LOG START option, 22 – 7
ARCHIVE LOG START option, s pecifying destination, 22 – 13
ARCHIVE LOG STOP option, 22 – 7
CHECKPOINT option, 5 – 13
ENABLE RESTRICTED SESSION option, 3 – 7
SET LICENSE_MAX_SESSIONS option, 19 – 4
SET LICENSE_MAX_USERS option, 19 – 5
SET LICENSE_SESSIONS_WARNING option, 19 – 4
SET MTS_DISPATCHERS option, 4 – 9
SET MTS_SERVERS option, 4 – 8
SET RESOURCE_LIMIT option, 19 – 19
SWITCH LOGFILE option, 5 – 13

ALTER TABLE command
ALLOCATE EXTENT option, 11 – 8
DISABLE ALL TRIGGERS option, 16 – 12
DISABLE integrity constraint option, 16 – 15
DROP integrity constraint option, 16 – 15
ENABLE ALL TRIGGERS option, 16 – 11
ENABLE integrity constraint option, 16 – 15
example, 11 – 8
MAXTRANS option, 10 – 10

ALTER TABLESPACE command
ADD DATAFILE parameter, 9 – 4
BEGIN BACKUP option
marking the beginning, 23 – 11
minimizing time between begin and end, 23 – 13
DEFAULT STORAGE option, 8 – 5
END BACKUP option
marking the end, 23 – 11
minimizing time between begin and end, 23 – 13
OFFLINE option, 23 – 14
ONLINE option
bringing tablespaces back online, 23 – 14
example, 8 – 8
READ ONLY option, 8 – 9 to 8 – 11
READ WRITE option, 8 – 11
RENAME DATA FILE option, 9 – 8

ALTER TRIGGER command
DISABLE option, 16 – 12
ENABLE option, 16 – 11

ALTER USER command, changing passwords with, 19 – 13

Alter User dialog, 19 – 12

ALTER USER privilege, 19 – 12

ALTER VIEW command, COMPILE option, 16 – 19

altering
cluster indexes, 14 – 9
clustered tables, 14 – 9
clusters, 14 – 8
database status, 3 – 6
hash clusters, 15 – 8
indexes, 13 – 8
passwords, 19 – 13
public rollback segments, 17 – 10
rollback segment storage parameters, 17 – 9
sequences, 12 – 11
storage parameters, 11 – 7
tables, 11 – 7, 11 – 8
tablespace storage, 8 – 5
users, 19 – 12

ANALYZE command
CASCADE option, 16 – 8
COMPUTE STATISTICS option, 16 – 6
ESTIMATE STATISTICS SAMPLE option, 16 – 6
LIST CHAINED ROWS option, 16 – 8
number of cluster keys, A – 13
shared SQL and, 16 – 7
STATISTICS option, 16 – 4
VALIDATE STRUCTURE option, 16 – 7

analyzing objects
about, 16 – 3
privileges, 16 – 3

application administrator, 1 – 3
database administrator versus, 18 – 10

application developers
privileges for, 18 – 9
roles for, 18 – 10

application development, security for, 18 – 9

ARCHIVE LOG command, LIST parameter, 5 – 9

archived redo log
applying during recovery, 24 – 11, 24 – 13
archiving modes, 22 – 5
automatic archiving, 22 – 6
automatic, disabling archiving, 22 – 7
default filename format, 22 – 12
deleting after recovery, 24 – 10
enabling automatic archiving, 22 – 6
errors during recovery, 24 – 15
filename format, 22 – 11
listing status, 22 – 10
location during recovery, 24 – 11
loss of, 24 – 52
preserving or resetting log sequence number, 24 – 26, 24 – 31, 24 – 35
restoring to disk, 24 – 9
setting archive destination, 22 – 13
specifying archive destination, 22 – 11

ARCHIVELOG mode
archiving, 22 – 2
datafile loss in, 24 – 48
distributed database backups, 23 – 4
enabling, 22 – 5
setting at database creation, 22 – 4
strategies for backups in, 23 – 6
taking datafiles offline and online in, 9 – 7

archiving
advantages, 22 – 2
automatic
disabling, 22 – 7
disabling after setup, 22 – 7
disabling at startup, 22 – 7
enabling, 22 – 6
changing mode, 22 – 5
default file name format, 22 – 12
disadvantages, 22 – 2
enabling and disabling, 22 – 4
enabling automatic, 22 – 6
increasing speed of, 22 – 10
manually, 22 – 8
minimizing impact on system performance, 22 – 9
privileges
disabling, 22 – 7
enabling, 22 – 6
for manually archiving, 22 – 8
to view status, 22 – 10
specifying destination, 22 – 11
tuning, 22 – 9
viewing information on, 22 – 10

AUDIT command, 21 – 12
schema objects, 21 – 13
statement auditing, 21 – 12
system privileges, 21 – 12

audit trail
See also auditing
archiving, 21 – 17
auditing changes to, 21 – 18

- controlling size of, 21 – 16
- creating and deleting, 21 – 4
- deleting views, 21 – 5
- interpreting, 21 – 18
- maximum size of, 21 – 16
- protecting integrity of, 21 – 18
- purging records from, 21 – 17
- recording changes to, 21 – 18
- records in, 21 – 7
- reducing size of, 21 – 17
- table that holds, 21 – 2
- views on, 21 – 5

AUDIT_TRAIL parameter, setting, 21 – 15

auditing
- *See also* audit trail
- AUDIT command, 21 – 12
- audit option levels, 21 – 8
- audit trail records, 21 – 6
- default options, 21 – 13
- disabling default options, 21 – 15
- disabling options, 21 – 14, 21 – 15
- disabling options versus auditing, 21 – 14
- enabling options, 21 – 12, 21 – 15
- enabling options versus auditing, 21 – 12
- guidelines, 21 – 2
- historical information, 21 – 4
- keeping information manageable, 21 – 2
- managing the audit trail, 21 – 4
- operating–system audit trails, 21 – 6
- policies for, 18 – 11
- privilege audit options, 21 – 10
- privileges required for object, 21 – 13
- privileges required for system, 21 – 13
- schema object types, 21 – 11
- schema objects, 21 – 13
- session level, 21 – 10
- shortcuts for object, 21 – 11
- shortcuts for system, 21 – 9
- statement, 21 – 12
- statement level, 21 – 8
- suspicious activity, 21 – 3
- system privileges, 21 – 12
- triggers and, 21 – 22
- using the database, 21 – 2
- viewing
  - active object options, 21 – 21
  - active privilege options, 21 – 21
  - active statement options, 21 – 21
  - default object options, 21 – 22
- views, 21 – 5

authentication
- changing, 19 – 13
- database managed, 19 – 8
- multi–threaded server and, 19 – 8
- operating system, 1 – 7
- operating–system managed, 19 – 7
- password file, 1 – 8
- password policy, 18 – 4
- specifying when creating a user, 19 – 9
- users, 18 – 2, 19 – 6, 19 – 7

authorization
- changing for roles, 20 – 11
- omitting for roles, 20 – 11
- operating–system role management and, 20 – 10
- roles
  - about, 20 – 9
  - multi–threaded server and, 20 – 10

automatic archiving, archive log destination, 22 – 6

B

background processes, Oracle7 processes, 4 – 11

BACKGROUND_DUMP_DEST parameter, 4 – 14

backups
- after creating new databases
  - full backups, 2 – 7
  - guidelines, 1 – 19
- after structural changes to database, 23 – 3
- ARCHIVELOG mode in, 23 – 6
- before database creation, 2 – 4
- checking datafile backup status, 23 – 12
- control files, 23 – 15
- creating a strategy, 23 – 5
- DB_VERIFY, 23 – 10
- DB_VERIFY utility, 23 – 10
- distributed databases, 23 – 4
- effects of archiving on, 22 – 2
- Export utility and, 23 – 4
- frequency of, 23 – 2
- full backups, about, 23 – 8

guidelines for, 23 – 2
keeping, 23 – 3
listing files needed, 23 – 8
marking end, 23 – 11
NOARCHIVELOG mode in, 23 – 5
often–used tablespaces, 23 – 3
partial backups, 23 – 10
planning before database creation, 23 – 2
privileges for control files, 23 – 15
procedures for
about, 23 – 7
offline datafiles, 23 – 14
offline tablespaces, 23 – 14
online datafiles, 23 – 10
online tablespaces, 23 – 10
read–only tablespaces, 23 – 7
restoring a full backup, 24 – 16
tablespaces, 23 – 13
test strategies, 23 – 2

Begin Manual Archive dialog box, 22 – 8

Begin Online Tablespace Backup dialog, 23 – 11

branch blocks, space required, A – 7

bringing online, tablespaces, 8 – 7

broken jobs
about, 7 – 11
marking, 7 – 11
running, 7 – 12

buffers, buffer cache in SGA, 2 – 11

bug fixes, 1 – 20

C

CASCADE option
integrity constraints, 14 – 11
when dropping unique or primary keys, 16 – 15

cascading revokes, 20 – 16

CATAUDIT.SQL, running, 21 – 5

CATBLOCK.SQL script, 4 – 10

CATNOAUD.SQL, running, 21 – 5

change–based recovery, procedure for, 24 – 24

CHAR datatype, 10 – 17
increasing column length, 11 – 7
space use of, 10 – 17

character sets
multi–byte characters
in role names, 20 – 8
in role passwords, 20 – 10
in user names, 19 – 9
user passwords and, 19 – 10
parameter file and, 3 – 10
specifying when creating a database, 2 – 2
supported by Oracle, 10 – 17

CHECK constraint, 16 – 14

checkpoint, 24 – 3

checkpoint process (CKPT), starting, 4 – 15

CHECKPOINT_PROCESS parameter, setting, 4 – 15

checkpoints
controlling, 5 – 10
fast checkpoint, 5 – 13
forcing, 5 – 13
log switches and, 5 – 10
setting intervals, 5 – 11

checksums
for data blocks, 9 – 11
for redo blocks, 5 – 14

CKPT. *See* checkpoint process (CKPT)

CLEAR LOGFILE clause, 5 – 14

clearing log files, 5 – 14

cluster keys
ANALYZE command and, A – 13
columns for, 14 – 4
SIZE parameter, 14 – 5

clustered tables, 14 – 10

clusters
allocating extents, 14 – 9
altering, 14 – 8
analyzing statistics, 16 – 3
choosing data, 14 – 4
columns for cluster key, 14 – 4
creating, 14 – 6
dropped tables and, 11 – 9
dropping, 14 – 9
estimating space, 14 – 5, A – 9 to A – 15
guidelines for managing, 14 – 3
hash, contrasted with index, 15 – 2
hash clusters, 15 – 1
index, contrasted with hash, 15 – 2

index creation, 14 – 7
indexes and, 13 – 2
keys, 14 – 2
location, 14 – 5
managing, 14 – 1
overview of, 14 – 2 to 14 – 6
privileges
for controlling, 20 – 7
for creating, 14 – 6
for dropping, 14 – 10
specifying PCTFREE for, 10 – 4
storage parameters, 10 – 11
truncating, 16 – 9
validating structure, 16 – 7

cold backups, full backups, 23 – 8

columns
displaying information about, 16 – 25
granting privileges for selected, 20 – 13
granting privileges on, 20 – 14
increasing length, 11 – 7
INSERT privilege and, 20 – 14
listing users granted to, 20 – 24
privileges, 20 – 14
revoking privileges on, 20 – 16

complete recovery, procedures for, 24 – 18

composite limits, 19 – 17
costs and, 19 – 18
service units, 19 – 17

COMPUTE STATISTICS option, 16 – 6

configuring an instance, with dedicated server processes, 4 – 2

CONNECT audit option, shortcut for auditing, 21 – 9

CONNECT role, 20 – 9

connecting
administrator privileges, 3 – 8
to a database as INTERNAL, 3 – 2

connections
auditing, 21 – 10
dedicated servers, 4 – 3
during shutdown, 3 – 8

control files
adding, 6 – 4
backing up, 23 – 8, 23 – 15
changing size, 6 – 4
conflicts with data dictionary, 6 – 8
creating
about, 6 – 3
additional control files, 6 – 4
initially, 6 – 4
new files, 6 – 5
default name, 2 – 10, 6 – 4
dropping, 6 – 9
during incomplete recovery, 24 – 25, 24 – 29, 24 – 33
errors during creation, 6 – 9
finding filenames, 23 – 8
guidelines for, 6 – 2
importance of mirrored, 6 – 2
location of, 6 – 3
loss of, 24 – 53, 24 – 54
managing, 6 – 1
mirroring, 2 – 10
moving, 6 – 4
names, 6 – 2
number of, 6 – 2
overwriting existing, 2 – 10
privileges to backup, 23 – 15
relocating, 6 – 4
renaming, 6 – 4
requirement of one, 6 – 3
size of, 6 – 3
specifying names before database creation, 2 – 10
unavailable during startup, 3 – 3
V$BACKUP view and, 23 – 12

CONTROL_FILES parameter
overwriting existing control files, 2 – 10
setting
before database creation, 2 – 10, 6 – 4
names for, 6 – 2

costs, resource limits and, 19 – 18

CREATE CLUSTER command
example, 14 – 6
for hash clusters, 15 – 5
HASH IS option, 15 – 6
HASHKEYS option, 15 – 7
SIZE option, 15 – 6, A – 12

CREATE CONTROLFILE command
about, 6 – 5
checking for inconsistencies, 6 – 8
NORESETLOGS option, 6 – 7
RESETLOGS option, 6 – 7

CREATE DATABASE command
  CONTROLFILE REUSE option, 6 – 4
  example, 2 – 7
  MAXDATAFILES option, 9 – 2
  MAXLOGFILES option, 5 – 4
  MAXLOGMEMBERS option, 5 – 4
CREATE INDEX command
  explicitly, 13 – 7
  ON CLUSTER option, 14 – 7
  temporary space required, A – 8
  UNRECOVERABLE, 13 – 5
  with a constraint, 13 – 7
CREATE PROFILE command
  about, 19 – 16
  COMPOSITE_LIMIT option, 19 – 17
Create Profile dialog, 19 – 16
CREATE ROLE command
  IDENTIFIED BY option, 20 – 10
  IDENTIFIED EXTERNALLY option, 20 – 10
CREATE ROLLBACK SEGMENT command
  about, 17 – 8
  tuning guidelines, 2 – 15
Create Rollback Segment dialog, 17 – 8
CREATE SCHEMA command
  multiple tables and views, 16 – 2
  privileges required, 16 – 2
CREATE SEQUENCE command, 12 – 10
CREATE SYNONYM command, 12 – 12
CREATE TABLE command
  about, 11 – 6
  CLUSTER option, 14 – 7
  UNRECOVERABLE, 11 – 4
CREATE TABLESPACE command
  datafile names in, 8 – 4
  example, 8 – 4
Create Tablespace dialog, 8 – 4
CREATE USER command
  IDENTIFIED BY option, 19 – 9
  IDENTIFIED EXTERNALLY option, 19 – 9
CREATE VIEW command
  about, 12 – 2
  OR REPLACE option, 12 – 8
  WITH CHECK OPTION, 12 – 3
creating
  audit trail, 21 – 4
  cluster index, 14 – 6
  clustered tables, 14 – 6
  clusters, 14 – 6
  control files, 6 – 3
  database, 1 – 19, 2 – 1
    archiving mode, 22 – 4
    backing up the new database, 2 – 7
    during installation, 2 – 3
    executing CREATE DATABASE, 2 – 6
    maximum number of datafiles, 9 – 2
    migration from different versions, 2 – 3
    preparing to, 2 – 2
    prerequisites for, 2 – 3
    problems encountered while, 2 – 8
    steps, 2 – 4
  datafiles, 8 – 3, 9 – 4
  hash clustered tables, 15 – 5
  hash clusters, 15 – 5
  indexes, explicitly, 13 – 7
  multiple objects, 16 – 2
  online redo log groups, 5 – 5
  parameter file, 2 – 4
  profiles, 19 – 16
  redo log members, 5 – 5
  rollback segments
    about, 17 – 8
    specifying storage parameters, 17 – 8
  sequences, 12 – 10
  synonyms, 12 – 12
  tables, 11 – 6
  tablespaces, 8 – 3
    rollback segments required, 8 – 4
  views, 12 – 2

D

data, security of, 18 – 3
data blocks
  altering size of, 2 – 11
  managing space usage of, 10 – 2
  managing space use of, 10 – 2
  operating system blocks versus, 2 – 11
  PCTFREE storage parameter, 10 – 3
  PCTUSED storage parameter, 10 – 5
  shared in clusters, 14 – 2
  size of, 2 – 11
  verifying, 9 – 11

data dictionary
  changing storage parameters, 16 – 23
  conflicts with control files, 6 – 8
  dropped tables and, 11 – 9
  schema object views, 16 – 23
  segments in the, 16 – 21
  setting storage parameters of, 16 – 21
  V$DBFILE view, 2 – 8
  V$DISPATCHER view, 4 – 9
  V$LOGFILE view, 2 – 8
  V$QUEUE view, 4 – 9
data integrity
  See also integrity constraints
  integrity constraints, 16 – 14
database administrator, 1 – 2
  application administrator versus, 18 – 10
  initial priorities, 1 – 17
  operating–system account, 1 – 4
  password files for, 1 – 7
  responsibilities of, 1 – 2
  roles
    about, 1 – 5
    for security, 18 – 7
  security and privileges of, 1 – 4
  security for, 18 – 7
  security officer versus, 1 – 3, 18 – 2
  usernames, 1 – 4
  utilities for, 1 – 16
database links
  job queues and, 7 – 8
  privileges for controlling, 20 – 7
databases
  administering, 1 – 1
  auditing, 21 – 1
  availability, 3 – 6
  backing up
    after creation of, 1 – 19
    full backups, 2 – 7
  checking the default instance, 2 – 6
  control files of, 6 – 2
  CREATE DATABASE command, 2 – 7
  creating
    about, 2 – 1
    opening and, 1 – 19
    steps for, 2 – 4
    trouble–shooting problems, 2 – 8
  design of, implementing, 1 – 19
  dropping, 2 – 8
  exclusive mode, 3 – 5
  global database name, about, 2 – 9
  global database names, in a distributed system, 2 – 9
  hardware evaluation, 1 – 18
  logical structure of, 1 – 18
  managing, size of, 9 – 1
  migration of, 2 – 3
  mounting a database, 3 – 3
  mounting to an instance, 3 – 6
  name, starting an instance, 3 – 2
  names
    about, 2 – 9
    conflicts in, 2 – 9
  opening, a closed database, 3 – 7
  parallel mode, 3 – 5
  physical structure of, 1 – 19
  planning, 1 – 18
  production, 18 – 9, 18 – 10
  renaming, 6 – 5
  restricting access to, 3 – 4, 3 – 7
  shutting down, 3 – 8
  specifying control files, 2 – 10
  starting up
    before database creation, 2 – 6
    general procedures for, 3 – 2
    parameter filenames, 3 – 3
    restricting access, 3 – 4
  structure of, distributed database, 1 – 19
  test, 18 – 9
  tuning
    archiving large databases, 22 – 9
    responsibilities for, 1 – 20
  user responsibilities, 1 – 3
  viewing datafiles and redo log files, 2 – 8
datafiles
  adding to a tablespace, 9 – 4
  backing up offline, 23 – 14
  backing up online, 23 – 10
  bringing online and offline, 9 – 6 to 9 – 7
  checking associated tablespaces, 8 – 14
  creating, 8 – 3
  database administrator's access to, 1 – 4
  default directory, 9 – 4
  dropping, 8 – 12
    NOARCHIVELOG mode, 9 – 7
  FILEXT$, 9 – 5
  fully specifying filenames, 9 – 4

identifying filenames, 9 – 10
location, 9 – 3
loss of, 24 – 48
managing, 9 – 1
maximum number of, 9 – 2
minimum number of, 9 – 2
MISSING, 6 – 8
monitoring, 9 – 12
offline, 9 – 7
online, 9 – 7
privileges to rename, 9 – 8
privileges to take offline, 9 – 7
recovering without a backup, 24 – 8
recovery creation, 24 – 8
relocating, 9 – 7 to 9 – 11
relocating, example, 9 – 9
renaming, 9 – 7 to 9 – 11
renaming for single tables, 9 – 8
reusing, 9 – 4
size of, 9 – 3
storing separately from redo log files, 9 – 3
unavailable when database is opened, 3 – 3
verifying data blocks, 9 – 11
viewing
backup status, 23 – 12
files needing recovery, 24 – 8
general status of, 9 – 12
V$DBFILE and V$LOGFILE views, 2 – 8

datatypes
character, 10 – 17
DATE, 10 – 18
individual type names, 10 – 17
LONG, 10 – 18
LONG RAW, 10 – 18
MLSLABEL, 10 – 19
NUMBER, 10 – 17
RAW, 10 – 18
ROWID, 10 – 18
space use of, 10 – 17
summarized, 10 – 20

DATE datatype, 10 – 18

DB_BLOCK_BUFFERS parameter, setting before database creation, 2 – 11

DB_BLOCK_CHECKSUM, 9 – 11

DB_BLOCK_SIZE parameter
database buffer cache size and, 2 – 11
setting before creation, 2 – 11

DB_DOMAIN parameter, setting before database creation, 2 – 9

DB_FILES parameter
default, 9 – 2
optional use of, 9 – 3

DB_NAME parameter
MTS_SERVICE and, 4 – 6
setting before database creation, 2 – 9

DBA. *See* database administrator

DBA audit option shortcuts, 21 – 9

DBA role, 1 – 5, 20 – 9

DBA_DATA_FILES, 8 – 13, 9 – 12

DBA_EXTENTS, 9 – 12

DBA_FREE_SPACE, 8 – 13, 9 – 12

DBA_FREE_SPACE_COALESCED view, 8 – 7

DBA_INDEXES view, filling with data, 16 – 5

DBA_ROLLBACK_SEGS view, 17 – 14, 17 – 15

DBA_SEGMENTS, 8 – 13, 9 – 12

DBA_TAB_COLUMNS view, filling with data, 16 – 5

DBA_TABLES view, filling with data, 16 – 5

DBA_TABLESPACES, 8 – 13, 9 – 12

DBA_TABLESPACES view, 8 – 12

DBA_TS_QUOTAS, 8 – 13, 9 – 12

DBA_USERS, 8 – 13, 9 – 12

DBMS_JOB package
altering a job, 7 – 10 to 7 – 13
forcing jobs to execute, 7 – 12
job queues and, 7 – 3
REMOVE procedure and, 7 – 9
submitting jobs, 7 – 4

DBMS_UTILITY.ANALYZE_SCHEMA(), running, 16 – 7

dedicated server processes
configuring, 4 – 2
connecting with, 4 – 3
trace files for, 4 – 12

dedicated servers, multi–threaded servers contrasted with, 4 – 3

default
archive log filename format, 22 – 12
audit options, 21 – 13
disabling, 21 – 15

profile, 19 – 16
role, 19 – 13
tablespace quota, 19 – 11
temporary tablespace, 19 – 10
user tablespaces, 19 – 10

deleting, table statistics, 16 – 4

dependencies, displaying, 16 – 26

developers, application, 18 – 9

disabling
archiving, 22 – 4
audit options, 21 – 14
auditing, 21 – 15
automatic archiving, 22 – 7
integrity constraints, 16 – 14
effects on indexes, 13 – 6
resource limits, 19 – 19
triggers, 16 – 11

disconnections, auditing, 21 – 10

dispatcher processes
calculating maximum number of, 4 – 7
number to start, 4 – 6
privileges to change number of, 4 – 9
removing, 4 – 9
service name, 4 – 6
setting the number of, 4 – 9
spawning new, 4 – 9

distributed databases
coordinated time–based recovery, 24 – 5
media recovery and snapshots, 24 – 6
recovery in, 24 – 5
running in ARCHIVELOG mode, 22 – 3
running in NOARCHIVELOG mode, 22 – 3
starting a remote instance, 3 – 5
taking backups, 23 – 4

distributed processing, parameter file location in, 3 – 11

distributing I/O, 2 – 16

DROP CLUSTER command
CASCADE CONSTRAINTS option, 14 – 11
dropping
cluster with no tables, 14 – 10
hash cluster, 15 – 9
INCLUDING TABLES option, 14 – 10

DROP INDEX command, 13 – 10

Drop Online Redo Log dialog, 5 – 9

Drop Online Redo Log Member dialog box, 5 – 10

DROP PROFILE command, 19 – 19

Drop Profile dialog, 19 – 19

DROP ROLE command, 20 – 11, 20 – 12

DROP ROLLBACK SEGMENT command, 17 – 14

Drop Rollback Segment dialog, 17 – 14

DROP SYNONYM command, 12 – 12

DROP TABLE command
about, 11 – 9
CASCADE CONSTRAINTS option, 11 – 9
for clustered tables, 14 – 10

DROP TABLESPACE command, 8 – 12

Drop Tablespace dialog box, 8 – 12

DROP USER command, 19 – 15

Drop User dialog, 19 – 15

DROP USER privilege, 19 – 15

dropping
audit trail, 21 – 4
cluster indexes, 14 – 9
clusters, 14 – 9
control files, 6 – 9
databases, 2 – 8
datafiles, 8 – 12
hash clusters, 15 – 9
indexes, 13 – 9
integrity constraints
about, 16 – 15
effects on indexes, 13 – 6
online redo log groups, 5 – 8
online redo log members, 5 – 9
profiles, 19 – 19
roles, 20 – 11
rollback segments, 17 – 11, 17 – 14
sequences, 12 – 11
synonyms, 12 – 12
tables, 11 – 9
tablespaces
about, 8 – 12
required privileges, 8 – 12
users, 19 – 15
views, 12 – 9

dynamic performance tables, using, 4 – 11

E

enabling
  archiving, 22 – 4
  auditing options
    about, 21 – 12
    privileges for, 21 – 15
  integrity constraints
    at creation, 16 – 14
    example, 16 – 14
    reporting exceptions, 16 – 16
    when violations exist, 16 – 13
  resource limits, 19 – 19
  triggers, 16 – 11

encryption, Oracle passwords, 19 – 8

End Online Tablespace Backup dialog, 23 – 11

enroll, database users, 1 – 20

environment of a job, 7 – 6

errors
  ALERT file and, 4 – 13
  during startup, 3 – 3
  ORA–00028, 4 – 17
  ORA–00114, 4 – 6
  ORA–01090, 3 – 8
  ORA–01173, 6 – 9
  ORA–01176, 6 – 9
  ORA–01177, 6 – 9
  ORA–1215, 6 – 9
  ORA–1216, 6 – 9
  ORA–1547, 16 – 23
  ORA–1628 through 1630, 16 – 23
  snapshot too old, 17 – 6
  trace files and, 4 – 12
  when creating a database, 2 – 8
  when creating control file, 6 – 9
  while starting an instance, 3 – 5

ESTIMATE STATISTICS option, 16 – 6

estimating size
  hash clusters, 15 – 4
  indexes, A – 5
  tables, 11 – 4, A – 4

evaluating, hardware for the Oracle7 Server, 1 – 18

example, creating constraints, 16 – 14

examples, altering an index, 13 – 8

exceptions, integrity constraints, 16 – 16

exclusive mode
  of the database, 3 – 5
  rollback segments and, 17 – 3
  terminating remaining user sessions, 4 – 16

EXP_FULL_DATABASE role, 20 – 9

Export utility
  about, 1 – 17
  backups and, 23 – 4
  read consistency and, 23 – 18
  restricted mode and, 3 – 4
  using for backup, 23 – 18

exporting jobs, 7 – 6

extents
  allocating
    clusters, 14 – 9
    index creation, 13 – 6
    tables, 11 – 8
  data dictionary views for, 16 – 24
  displaying free extents, 16 – 27
  displaying information on, 16 – 26
  dropped tables and, 11 – 9

F

failures
  media (disk), 24 – 47
  media failure, 24 – 47

fast checkpoint, 5 – 13

filenames, listing, 23 – 8

files, OS limit on number open, 8 – 2

FILEXT$, 9 – 5

Force Checkpoint menu option, 5 – 13

Force Log Switch menu option, 5 – 13

FOREIGN KEY constraint, enabling, 16 – 14

fragmentation, reducing, 10 – 9

free space
  coalescing, 8 – 6
  listing free extents, 16 – 27
  tablespaces and, 8 – 14

full backups, 23 – 8
  restoring, 24 – 16

functions, recompiling, 16 – 19

G

global database name, 2 – 9

GRANT command
- ADMIN option, 20 – 12
- GRANT option, 20 – 14
- object privileges, 20 – 13
- SYSOPER/SYSDBA privileges, 1 – 13
- system privileges and roles, 20 – 12
- when takes effect, 20 – 18

GRANT OPTION
- about, 20 – 14
- revoking, 20 – 15

granting privileges and roles
- listing grants, 20 – 22
- shortcuts for object privileges, 20 – 7
- SYSOPER/SYSDBA privileges, 1 – 13

group, redo log, online redo log, archived redo log, 24 – 49

guidelines, for managing rollback segments, 17 – 2

H

hardware, evaluating, 1 – 18

hash clusters
- altering, 15 – 8
- choosing key, 15 – 5
- clusters, 15 – 1
- controlling space use of, 15 – 5
- creating, 15 – 5
- dropping, 15 – 9
- estimating storage, 15 – 4
- example, 15 – 7
- managing, 15 – 1
- usage, 15 – 2

high water mark, for a session, 19 – 3

HOST, command in Server Manager, 5 – 7

hot backups, partial backups, 23 – 10

I

I/O, distributing, 2 – 16

identification, users, 19 – 6

IMP_FULL_DATABASE role, 20 – 9

implementing database design, 1 – 19

Import utility
- about, 1 – 17
- procedure for using, 23 – 19
- restricted mode and, 3 – 4
- using for recovery, 23 – 18

importing, jobs, 7 – 6

in–doubt transactions, rollback segments and, 17 – 11

incomplete recovery, procedures for, 24 – 24

indexes
- altering, 13 – 8
- analyzing statistics, 16 – 3
- cluster
  - altering, 14 – 9
  - creating, 14 – 6
  - dropping, 14 – 9
  - managing, 14 – 1
- correct tables and columns, 13 – 6
- creating
  - after inserting table data, 13 – 3
  - explicitly, 13 – 7
  - unrecoverably, 13 – 5
- disabling and dropping constraints and, 13 – 6
- dropped tables and, 11 – 9
- dropping, 13 – 9
- estimating size, 13 – 5, A – 5 to A – 10
- extent allocation for, 13 – 6
- guidelines for managing, 13 – 2
- INITRANS for, 13 – 4
- limiting per table, 13 – 3
- managing, 13 – 1, 13 – 9
- MAXTRANS for, 13 – 4
- monitoring space use of, 13 – 9
- overview of, 13 – 2 to 13 – 6
- parallelizing index creation, 13 – 4
- PCTFREE for, 13 – 4
- PCTUSED for, 13 – 4
- privileges
  - for altering, 13 – 8
  - for controlling, 20 – 7
  - for creating, 13 – 6
  - for dropping, 13 – 9
- separating from a table, 11 – 5
- setting storage parameters for, 13 – 5
- SQL*Loader and, 13 – 3

storage parameters, 10 – 11
tablespace for, 13 – 4
temporary segments and, 13 – 3
temporary space and, A – 8
validating structure, 16 – 7

initial, passwords for SYS and SYSTEM, 1 – 4

INITIAL storage parameter, 10 – 8
altering, 11 – 7

initialization parameters
affecting sequences, 12 – 11
multi–threaded server and, 4 – 4

INITRANS storage parameter
altering, 11 – 7
default, 10 – 10
guidelines for setting, 10 – 10
transaction entries and, 10 – 10

INSERT privilege
granting, 20 – 14
revoking, 20 – 16

installation
and creating a database, 2 – 3
Oracle7 Server, 1 – 18
tuning recommendations for, 2 – 14

instance identifier
checking, 2 – 6
process names and, 4 – 12

instance menu
Open option, 3 – 7
prevent Connections option, 3 – 7

instances
aborting, 3 – 9
default, 2 – 6
shutting down immediately, 3 – 9
starting, 3 – 2
starting before database creation, 2 – 6

integrity constraints
disabling, 16 – 12, 16 – 14
disabling on creation, 16 – 14
dropping, 16 – 15
dropping and disabling, 13 – 6
dropping tablespaces and, 8 – 12
enabling, 16 – 12
enabling on creation, 16 – 14
enabling when violations exist, 16 – 13
exceptions to, 16 – 16
managing, 16 – 12
UTLEXCPT.SQL script, 16 – 16
violations, 16 – 13
when to disable, 16 – 13

INTERNAL
alternatives to, 1 – 7
connecting for shutdown, 3 – 8
creating a database as, 2 – 6
OSOPER and OSDBA, 1 – 7
security for, 18 – 7
starting an instance as, 3 – 2

INTERNAL date function, executing jobs and, 7 – 7

interrupting, media recovery, 24 – 15

J

job queues, 7 – 2
executing jobs in, 7 – 8
locks, 7 – 8
multiple SNP processes and, 7 – 2
privileges for using, 7 – 4
removing jobs from, 7 – 9
scheduling jobs, 7 – 3
starting up SNP processes, 7 – 3
submitting new jobs, 7 – 4
viewing, 7 – 13 to 7 – 15

jobs
altering, 7 – 10 to 7 – 13
broken, 7 – 11
database links and, 7 – 8
executing, 7 – 8
exporting, 7 – 6
forcing to execute, 7 – 12
importing, 7 – 6
INTERNAL date function and, 7 – 7
job definition, 7 – 7
job number, 7 – 6
killing, 7 – 13
managing, 7 – 3
marking broken jobs, 7 – 11
ownership of, 7 – 6
removing from job queue, 7 – 9
running broken jobs, 7 – 12
scheduling, 7 – 3
submitting to job queue, 7 – 4
trace files, 7 – 9
troubleshooting, 7 – 9

join view, 12 – 4
DELETE statements, 12 – 7
key–preserved tables in, 12 – 5
mergeable, 12 – 5
modifying, rule for, 12 – 6
when modifiable, 12 – 4
JQ locks, 7 – 8

K

key–preserved tables, in join views, 12 – 5
keys, cluster, 14 – 2
Kill User Session dialog, 4 – 16
killing, jobs, 7 – 13

L

LGWR. *See* log writer process (LGWR)
LICENSE_MAX_SESSIONS parameter
changing while instance runs, 19 – 4
setting, 19 – 3
setting before database creation, 2 – 12
LICENSE_MAX_USERS parameter
changing while database runs, 19 – 5
setting, 19 – 5
setting before database creation, 2 – 12
LICENSE_SESSION_WARNING parameter, setting before database creation, 2 – 12
LICENSE_SESSIONS_WARNING parameter
changing while instance runs, 19 – 4
setting, 19 – 4
licensing
complying with license agreement, 2 – 12, 19 – 2
concurrent usage, 19 – 2
named user, 19 – 2, 19 – 4
number of concurrent sessions, 2 – 12
privileges for changing named user limits, 19 – 5
privileges for changing session limits, 19 – 4
session–based, 19 – 2
viewing limits, 19 – 5
limits
composite limits, 19 – 17
concurrent usage, 19 – 2
resource limits, 19 – 17
session, high water mark, 19 – 3
LIST CHAINED ROWS option, 16 – 8
listener process
configuration file, 4 – 5
setting MTS_LISTENER_ADDRESS, 4 – 5
location, rollback segments, 17 – 7
locks
job queue, 7 – 8
monitoring, 4 – 10
log sequence numbers, 5 – 2
See also redo log files
preserving after recovery, 24 – 26, 24 – 31, 24 – 35
requested during recovery, 24 – 11
resetting to 1, 24 – 26, 24 – 31, 24 – 35
log switches
checkpoints and, 5 – 10
forcing, 5 – 12
privileges, 5 – 13
log writer process (LGWR), trace file monitoring, 4 – 13
LOG_ARCHIVE_BUFFER_SIZE parameter, setting, 22 – 9, 22 – 10
LOG_ARCHIVE_BUFFERS parameter, setting, 22 – 9, 22 – 10
LOG_ARCHIVE_DEST parameter, setting, 22 – 13
LOG_ARCHIVE_FORMAT parameter, setting, 22 – 11
LOG_ARCHIVE_START parameter, 22 – 6
setting, 22 – 6, 22 – 7
LOG_CHECKPOINT_INTERVAL parameter, setting, 5 – 11
LOG_CHECKPOINT_TIMEOUT parameter, setting, 5 – 12
LOG_FILES parameter, number of log files and, 5 – 4
logical structure of a database, 1 – 18
LONG datatype, 10 – 18
LONG RAW datatype, 10 – 18

M

maintenance release number. *See* releases

managing
  auditing, 21 – 1
  cluster indexes, 14 – 1
  clustered tables, 14 – 1
  clusters, 14 – 1
  indexes, 13 – 1, 13 – 9
  jobs, 7 – 3
  object dependencies, 16 – 18
  profiles, 19 – 16
  roles, 20 – 8
  rollback segments, 17 – 1
  sequences, 12 – 10
  synonyms, 12 – 12
  tables, 11 – 1
  users, 19 – 8
  views, 12 – 1, 12 – 9

manual archiving, in ARCHIVELOG mode, 22 – 8

marked user session, 4 – 17

MAX_DUMP_FILE_SIZE parameter, 4 – 14

MAX_ENABLED_ROLES parameter
  default roles and, 20 – 11
  enabling roles and, 20 – 11

MAXDATAFILES parameter
  changing, 6 – 5
  optional use of, 9 – 3

MAXEXTENTS storage parameter
  about, 10 – 8
  setting for the data dictionary, 16 – 21

MAXINSTANCES parameter, changing, 6 – 5

MAXLOGFILES parameter
  changing, 6 – 5
  number of log files and, 5 – 4

MAXLOGHISTORY, changing, 6 – 5

MAXLOGMEMBERS parameter
  changing, 6 – 5
  number of log files and, 5 – 4

MAXTRANS storage parameter
  altering, 11 – 7
  default, 10 – 10
  guidelines for setting, 10 – 10
  transaction entries and, 10 – 10

media failure (disk failure)
  about, 24 – 47
  archived redo log file loss, 24 – 52
  control file loss, 24 – 53, 24 – 54
  datafile loss, 24 – 48
  online redo log group loss, 24 – 49
  online redo log loss, 24 – 49

media recovery
  applying archived redo logs, 24 – 11
  at instance startup, 24 – 11
  cancel–based, 24 – 24
  change–based, 24 – 24
  complete recovery, 24 – 18
  completion of, 24 – 19, 24 – 21, 24 – 23
  coordinated time–based distributed database, 24 – 5
  deciding which files need recovery, 24 – 8
  determining appropriate operation, 24 – 3
  distributed databases, 24 – 5
  errors with redo log files, 24 – 15
  guidelines and prerequisites, 24 – 3
  incomplete, 24 – 24
  interrupting, 24 – 15
  lost files
    about, 24 – 47 to 24 – 56
    lost archived redo log files, 24 – 52
    lost control files, 24 – 53
    lost datafiles, 24 – 48
    lost online redo log files, 24 – 49
  lost mirrored control files, 24 – 53
  NOARCHIVELOG mode, 24 – 16
  open database–offline tablespace, 24 – 20
  performing, about, 24 – 7
  requirements of
    incomplete, 24 – 26, 24 – 31, 24 – 35
  restoring
    archived redo log files, 24 – 9
    damaged files, 24 – 8
    full backups, 24 – 16
    to different devices, 24 – 4
  resuming, 24 – 15
  Server Manager and, 24 – 12
  snapshots and, 24 – 6
  starting, 24 – 10
  successfully applied redo logs, 24 – 15
  SYSTEM tablespace, 24 – 20, 24 – 22
  testing, 24 – 3

time–based, 24 – 24
undamaged tablespaces
online, 24 – 10, 24 – 20
memory, viewing per user, 19 – 23
migration, database migration, 2 – 3
MINEXTENTS storage parameter
about, 10 – 9
altering, 11 – 7
mirrored control files
importance of, 6 – 2
loss of, 24 – 53
mirrored redo log files
location of, 5 – 3
size of, 5 – 3
mirroring, control files, 2 – 10
MLSLABEL datatype, 10 – 19
modes
exclusive, 3 – 5
parallel, 3 – 5
restricted, 3 – 4, 3 – 7
modifiable join view, definition of, 12 – 4
modifying, a join view, 12 – 4
MONITOR command, ROLLBACK option, 17 – 15
monitoring
datafiles, 9 – 12
locks, 4 – 10
performance tables, 4 – 11
processes of an instance, 4 – 10
rollback segments, 17 – 6, 17 – 15
tablespaces, 9 – 12
mounting a database, 3 – 3
exclusive mode, 3 – 5
parallel mode, 3 – 5
moving
control files, 6 – 4
relocating, 9 – 7 to 9 – 11
MTS_DISPATCHERS parameter, setting initially, 4 – 6
MTS_LISTENER_ADDRESS parameter
setting, 4 – 5
starting new dispatchers and, 4 – 9
MTS_MAX_DISPATCHERS parameter, 4 – 7
setting, 4 – 7
MTS_MAX_SERVERS parameter, setting, 4 – 8
MTS_SERVERS parameter
minimum value, 4 – 8
setting, 4 – 7
MTS_SERVICE parameter
DB_NAME parameter as default, 4 – 6
setting, 4 – 6
multiplex online redo logs, symmetric groups, 5 – 2
multiplexing
online redo log, 5 – 2
redo log files, 5 – 2
See also redo log files, mirrored
multiplexing online redo log, 5 – 2
multi–threaded server
configuring dispatchers, 4 – 6
database shutdown and, 3 – 8
database startup and, 3 – 2
dedicated server contrasted with, 4 – 3
enabling and disabling, 4 – 8
operating–system authentication restrictions, 19 – 8
OS role management restrictions, 20 – 21
restrictions on OS role authorization, 20 – 10
service name, 4 – 6
shared pool and, 4 – 5
starting, 4 – 4

N

name resolution, 16 – 20
named user limits, 19 – 4
setting initially, 2 – 13
network protocol, dispatcher for each, 4 – 6
NEXT storage parameter, 10 – 8
setting for the data dictionary, 16 – 21
NOARCHIVELOG mode
archiving, 22 – 2
datafile loss, 24 – 48
distributed database backups, 23 – 4
partial backups and, 23 – 10
setting at database creation, 22 – 4
strategies for backups in, 23 – 5
taking datafiles offline in, 9 – 7
NOAUDIT command
disabling audit options, 21 – 14

privileges, 21 – 14
schema objects, 21 – 14
statements, 21 – 14
non–clustered tables, estimating size of, A – 2
NORESET LOGS option, backing up control file, 23 – 17
NOT NULL constraint, 16 – 14
NUMBER datatype, 10 – 17

O

objects, schema
cascading effects on revoking, 20 – 17
default tablespace for, 19 – 10
granting privileges, 20 – 13
in a revoked tablespace, 19 – 11
owned by dropped users, 19 – 15
privileges with, 20 – 6
revoking privileges, 20 – 15
offline backups, 23 – 2
offline datafiles, 9 – 7
offline rollback segments
about, 17 – 10
bringing online, 17 – 11
when to use, 17 – 11
offline tablespaces
altering, 8 – 7
priorities, 8 – 8
rollback segments and, 17 – 11
online backups, 23 – 2
online datafiles, 9 – 7
online redo log
See also redo log files, online
active group, 24 – 49
applying during recovery, 24 – 13
archived group, 24 – 49
creating groups, 5 – 5
creating members, 5 – 5
current group, 24 – 49
dropping groups, 5 – 8
dropping members, 5 – 9
forcing a log switch, 5 – 12
guidelines for configuring, 5 – 2
inactive group, 24 – 49
location of, 5 – 3
loss of, 24 – 49
loss of all members, 24 – 49
loss of group, 24 – 49
loss of mirrored members, 24 – 49
managing, 5 – 1
moving files, 5 – 8
multiplexing, 5 – 2
number of files, 5 – 4
preserving or resetting log sequence number, 24 – 26, 24 – 31, 24 – 35
privileges
adding groups, 5 – 5
dropping groups, 5 – 8
dropping members, 5 – 9
forcing a log switch, 5 – 13
renaming files, 5 – 8
renaming members, 5 – 6
STALE members, 5 – 10
status of members, 24 – 49
storing separately from datafiles, 9 – 3
unavailable when database is opened, 3 – 3
viewing filenames, 23 – 8
viewing information about, 5 – 15
online rollback segments
about, 17 – 10
bringing rollback segments online, 17 – 11
taking offline, 17 – 12
when new, 17 – 8
online tablespaces, altering, 8 – 7
opening a database
after creation, 1 – 19
mounted database, 3 – 7
operating system, 2 – 5
accounts, 20 – 20
auditing with, 21 – 2
authentication, 20 – 18
database administrator’s requirements for, 1 – 4
deleting datafiles, 8 – 12
enabling and disabling roles, 20 – 21
limit of number of open files, 9 – 2
Oracle7 process names, 4 – 11
renaming and relocating files, 9 – 8
role identification, 20 – 20
role management, 20 – 21
roles and, 20 – 18
security in, 18 – 3
OPS$ account, OS_AUTHENT_PREFIX parameter, 19 – 7

OPTIMAL storage parameter, 17 – 6
options of Oracle7 Server, i
Oracle blocks. *See* data blocks
Oracle Parallel Server. *See* Parallel Server
Oracle7 Server
  complying with license agreement, 19 – 2
  identifying releases, 1 – 20
  installing, 1 – 18
  options, i
  processes
    checkpoint (CKPT), 4 – 15
    monitoring, 4 – 10
    operating–system names, 4 – 11
    service names for dispatchers, 4 – 6
    trace files, 4 – 12
Oracle7 Server processes, processes
  dedicated server processes, 4 – 2
  identifying and managing, 4 – 10
ORAPWD utility, 1 – 9 to 1 – 16
OS authentication, 1 – 7
OS_AUTHENT_PREFIX parameter
  operating–system authentication and, 19 – 7
  setting, 19 – 7
OS_ROLES parameter
  operating–system authorization and, 20 – 10
  REMOTE_OS_ROLES and, 20 – 21
  using, 20 – 20
OSDBA, privileges included in, 1 – 8
OSOPER, privileges included in, 1 – 7
owner of a queued job, 7 – 6

P

packages
  privileges for recompiling, 16 – 19
  recompiling, 16 – 19
parallel mode, of the database, 3 – 5
parallel query option
  number of server processes, 4 – 15
  parallelizing index creation, 13 – 4
  parallelizing table creation, 11 – 3
  query servers, 4 – 15
Parallel Server
  ALTER CLUSTER..ALLOCATE EXTENT, 14 – 9
  archive log file name format, 22 – 12
  datafile upper bound for instances, 9 – 3
  forcing a checkpoint for the local instance, 5 – 13
  licensed session limit and, 2 – 13
  limits on named users and, 19 – 5
  LOG_CHECKPOINT_TIMEOUT and, 5 – 12
  managing rollback segments, 17 – 1
  named users and, 2 – 13
  own rollback segments, 17 – 3
  sequence numbers and, 12 – 10
  session and warning limits, 19 – 3
  specifying thread for archiving, 22 – 8
  switching archiving modes, 22 – 5
  V$THREAD view, 5 – 15
PARALLEL_MAX_SERVERS parameter, 4 – 15
  parallel recovery, 24 – 17
PARALLEL_MIN_SERVERS parameter, 4 – 16
PARALLEL_SERVER_IDLE_TIME parameter, 4 – 16
parameter files
  character set of, 3 – 10
  choosing for startup, 3 – 3
  creating for database creation, 2 – 4
  default for instance startup, 3 – 3
  editing, 3 – 10
  editing before database creation, 2 – 5
  individual parameter names, 2 – 9
  location of, 3 – 11
  minimum set of, 2 – 9
  number of, 3 – 10
  sample of, 3 – 10
  using, 3 – 10
partial backups, 23 – 10
passwords
  altering user passwords, 19 – 13
  authentication file for, 1 – 8
  changing for roles, 20 – 11
  initial for SYS and SYSTEM, 1 – 4
  password file
    adding users to, 1 – 12
    creating, 1 – 9 to 1 – 16
    OS authentication, 1 – 7
    relocating, 1 – 15
    removing, 1 – 15
    state of, 1 – 16
  privileges for changing for roles, 20 – 9

privileges to alter, 19 – 12
roles, 20 – 10
security policy for users, 18 – 4
setting REMOTE_LOGIN_PASSWORD parameter, 1 – 11
user authentication, 19 – 8

patch release number. *See* releases

PCTFREE storage parameter
altering, 11 – 7
block overhead and, 10 – 6
clustered tables, 10 – 4
default, 10 – 3
guidelines for setting, 10 – 3
how it works, 10 – 2 to 10 – 4
indexes, 10 – 4
non–clustered tables, 10 – 4
PCTUSED and, 10 – 6

PCTINCREASE storage parameter
about, 10 – 9
altering, 10 – 11
setting for the data dictionary, 16 – 21

PCTUSED storage parameter
altering, 11 – 7
block overhead and, 10 – 6
default, 10 – 5
guidelines for setting, 10 – 5
how it works, 10 – 4 to 10 – 7
PCTFREE and, 10 – 6

performance
location of datafiles and, 9 – 3
tuning archiving, 22 – 9

performance tables, dynamic performance tables, 4 – 11

physical structure of a database, 1 – 19

PL/SQL program units
dropped tables and, 11 – 9
replaced views and, 12 – 9

planning
database creation, 2 – 2
relational design, 1 – 19
the database, 1 – 18

precedence of storage parameters, 10 – 12

predefined roles, 1 – 5

prerequisites, for creating a database, 2 – 3

PRIMARY KEY constraint
disabling, 16 – 14
dropping associated indexes, 13 – 10
enabling, 16 – 14
enabling on creation, 13 – 6
foreign key references when dropped, 16 – 15
indexes associated with, 13 – 6
storage of associated indexes, 13 – 7

private
rollback segments, 17 – 8
taking offline, 17 – 12
synonyms, 12 – 12

privileges, 20 – 2
See also revoking privileges
adding datafiles to a tablespace, 9 – 4
adding redo log groups, 5 – 5
altering
default storage parameters, 8 – 5
dispatcher privileges, 4 – 9
indexes, 13 – 8
named user limit, 19 – 5
passwords, 19 – 13
role authentication, 20 – 9
rollback segments, 17 – 10
sequences, 12 – 10
tables, 11 – 7
users, 19 – 12
analyzing objects, 16 – 3
application developers and, 18 – 9
audit object, 21 – 13
auditing system, 21 – 13
auditing use of, 21 – 10
backing up, control files, 23 – 15
bringing datafiles offline and online, 9 – 7
bringing tablespaces online, 8 – 7
cascading revokes, 20 – 16
cluster creation, 14 – 6
coalescing tablespaces, 8 – 6
column, 20 – 14
CREATE SCHEMA command, 16 – 2
creating
indexes, 13 – 6
roles, 20 – 8
rollback segments, 17 – 8
sequences, 12 – 10
synonyms, 12 – 12
tables, 11 – 6
tablespaces, 8 – 4

users, 19 – 9
views, 12 – 2
database administrator, 1 – 4
disabling automatic archiving, 22 – 7
dropping
clusters, 14 – 10
indexes, 13 – 9
online redo log members, 5 – 9
redo log groups, 5 – 8
roles, 20 – 12
rollback segments, 17 – 14
sequences, 12 – 11
synonyms, 12 – 12
tables, 11 – 9
views, 12 – 9
dropping profiles, 19 – 19
enabling and disabling resource limits, 19 – 19
enabling and disabling triggers, 16 – 11
enabling automatic archiving, 22 – 6
for changing session limits, 19 – 4
forcing a checkpoint, 5 – 13
forcing a log switch, 5 – 13
granting
about, 20 – 12
object privileges, 20 – 13
required privileges, 20 – 13
system privileges, 20 – 12
grouping with roles, 20 – 8
individual privilege names, 20 – 2
job queues and, 7 – 4
listing grants, 20 – 23
manually archiving, 22 – 8
object, 20 – 6
on selected columns, 20 – 16
operating system, required for database administrator, 1 – 4
policies for managing, 18 – 5
recompiling packages, 16 – 19
recompiling procedures, 16 – 19
recompiling views, 16 – 19
renaming
datafiles of a tablespace, 9 – 8
datafiles of several tablespaces, 9 – 9
objects, 16 – 3
redo log members, 5 – 6
replacing views, 12 – 8
RESTRICTED SESSION system privilege, 3 – 4, 3 – 7
revoking, 20 – 15
ADMIN OPTION, 20 – 15
GRANT OPTION, 20 – 15
object privileges, 20 – 17
system privileges, 20 – 15
revoking object, 20 – 15
revoking object privileges, 20 – 15
setting resource costs, 19 – 18
SQL statements permitted by, 20 – 7
system, 20 – 2
taking rollback segments online and offline, 17 – 10
taking tablespaces offline, 8 – 8
truncating, 16 – 10
viewing archive status, 22 – 10

procedures, recompiling, 16 – 19

processes, 4 – 1
SNP background processes, 7 – 2

PROCESSES parameter, setting before database creation, 2 – 12

profiles, 19 – 16
altering, 19 – 17
assigning to users, 19 – 17
composite limit, 19 – 17
creating, 19 – 16
default, 19 – 16
disabling resource limits, 19 – 19
dropping, 19 – 19
enabling resource limits, 19 – 19
listing, 19 – 20
managing, 19 – 16
privileges for dropping, 19 – 19
privileges to alter, 19 – 17
privileges to set resource costs, 19 – 18
PUBLIC_DEFAULT, 19 – 16
setting a limit to null, 19 – 17
viewing, 19 – 22

program global area (PGA), effect of MAX_ENABLED_ROLES on, 20 – 11

pseudo–column, 10 – 18

public, synonyms, 12 – 12

public rollback segments
making available for use, 17 – 10
taking offline, 17 – 12

PUBLIC user group
granting and revoking privileges to, 20 – 18
procedures and, 20 – 18

PUBLIC_DEFAULT profile
dropping profiles and, 19 – 19
using, 19 – 16

Q

query server process, about, 4 – 15
quotas
listing, 19 – 20
revoking from users, 19 – 11
setting to zero, 19 – 11
tablespace, 19 – 11
tablespace quotas, 8 – 3
temporary segments and, 19 – 11
unlimited, 19 – 11
viewing, 19 – 21, 19 – 22

R

RAW datatype
about, 10 – 18
VARCHAR2 and, 10 – 18
read consistency, Export utility and, 23 – 18
read–only tablespaces
altering to writeable, 8 – 11
backing up, 23 – 7
creating, 8 – 9 to 8 – 11
datafiles, 9 – 6 to 9 – 7
on a WORM device, 8 – 11
recovering, 24 – 46 to 24 – 47
recompiling
automatically, 16 – 19
functions, 16 – 19
packages, 16 – 19
procedures, 16 – 19
views, 16 – 19
Recover Closed Database dialog box, 24 – 10
RECOVER command
DATABASE parameter, 24 – 10
DATAFILE parameter, 24 – 11
TABLESPACE parameter, 24 – 10
Recover Offline Tablespaces dialog box, 24 – 10
recovery
creating new control files, 6 – 5
effects of archiving on, 22 – 2
Import utility, 23 – 19
media recovery, 24 – 1
PARALLEL_MAX_SERVERS
parameter, 24 – 17
read–only tablespaces, 24 – 46 to 24 – 47
setting parallelism, 24 – 17
startup with automatic, 3 – 5
RECOVERY_PARALLELISM
parameter, 24 – 17
redo log
archived redo log, 22 – 2
online redo log, 5 – 1
relocating, 5 – 7
renaming, 5 – 7
redo log files
archived redo log files, 22 – 4
log sequence numbers of, defined, 5 – 2
multiplexing, overview of, 5 – 2
online, 5 – 2
viewing, 2 – 8
redo records, 24 – 2
reducing fragmentation, 10 – 9
REFERENCES privilege
CASCADE CONSTRAINTS option, 20 – 16
revoking, 20 – 16
relational design, planning, 1 – 19
releases
checking the release number, 1 – 21
identifying for Oracle7 Server, 1 – 20
maintenance release number, 1 – 21
patch release number, 1 – 21
port–specific release number, 1 – 21
versions of other Oracle software, 1 – 21
relocating
control files, 6 – 4
datafiles, 9 – 7 to 9 – 11
remote connections
connecting as INTERNAL, 1 – 14
connecting as SYSOPER/SYSDBA, 1 – 14
password files, 1 – 9 to 1 – 16
REMOTE_LOGIN_PASSWORDFILE
parameter, 1 – 11
REMOTE_OS_AUTHENT parameter,
setting, 19 – 8
REMOTE_OS_ROLES parameter,
setting, 20 – 10, 20 – 21
RENAME command, 16 – 3

Rename Data File dialog, 9 – 8

Rename Online Redo Log Member dialog box, 5 – 7

renaming
- control files, 6 – 4
- datafiles, 9 – 7 to 9 – 11
- datafiles with a single table, 9 – 8
- online redo log members, 5 – 6
- schema objects, 16 – 3

replacing, views, 12 – 8

RESOURCE audit option, shortcut for auditing, 21 – 9

resource limits
- altering in profiles, 19 – 17
- assigning with profiles, 19 – 17
- composite limits and, 19 – 17
- costs and, 19 – 18
- creating profiles and, 19 – 16
- disabling, 19 – 19
- enabling, 19 – 19
- privileges to enable and disable, 19 – 19
- privileges to set costs, 19 – 18
- profiles, 19 – 16
- PUBLIC_DEFAULT profile and, 19 – 16
- service units, 19 – 17
- setting to null, 19 – 17

RESOURCE role, 20 – 9

RESOURCE_LIMIT parameter, enabling and disabling limits, 19 – 19

resources, profiles, 19 – 16

responsibilities
- of a database administrator, 1 – 2
- of database users, 1 – 3

restoring, full backup, 24 – 16

RESTRICTED SESSION privilege
- instances in restricted mode, 3 – 7
- restricted mode and, 3 – 4
- session limits and, 19 – 2

restricting access to database, starting an instance, 3 – 4

resuming recovery, 24 – 15

REVOKE command, 20 – 15
- when takes effect, 20 – 18

Revoke System Privileges / Roles dialog, 20 – 15

revoking, privileges and roles, SYSOPER / DBA privileges, 1 – 13

revoking privileges and roles
- on selected columns, 20 – 16
- REVOKE command, 20 – 15
- shortcuts for object privileges, 20 – 7
- when using operating–system roles, 20 – 21

roles
- ADMIN OPTION and, 20 – 12
- application developers and, 18 – 10
- authorization, 20 – 9
- backward compatibility, 20 – 9
- changing authorization for, 20 – 11
- changing passwords, 20 – 11
- CONNECT role, 20 – 9
- database authorization, 20 – 10
- DBA role, 1 – 5, 20 – 9
- default, 19 – 13
- dropping, 20 – 11
- EXP_FULL_DATABASE, 20 – 9
- GRANT command, 20 – 21
- GRANT OPTION and, 20 – 14
- granting, about, 20 – 12
- grouping with roles, 20 – 8
- IMP_FULL_DATABASE, 20 – 9
- listing, 20 – 25
- listing grants, 20 – 24
- listing privileges and roles in, 20 – 26
- management using the operating system, 20 – 18
- managing, 20 – 8
- multi–byte characters, in names, 20 – 8
- multi–byte characters in passwords, 20 – 10
- multi–threaded server and, 20 – 10
- operating system granting of, 20 – 20, 20 – 21
- operating–system authorization, 20 – 10
- OS management and the multi–threaded server, 20 – 21
- passwords for enabling, 20 – 10
- predefined, 1 – 5, 20 – 9
- privileges
  - changing authorization method, 20 – 9
  - changing passwords, 20 – 9
  - for creating, 20 – 8
  - for dropping, 20 – 12
  - granting system privileges or roles, 20 – 12

RESOURCE role, 20 – 9
REVOKE command, 20 – 21
revoking, 20 – 15
revoking ADMIN OPTION, 20 – 15
security and, 18 – 5
SET ROLE command, 20 – 21
unique names for, 20 – 8
without authorization, 20 – 11

rollback segments
acquiring automatically, 17 – 4, 17 – 12
acquiring on startup, 2 – 12
allocating, 2 – 15
altering public, 17 – 10
altering storage parameters, 17 – 9
AVAILABLE, 17 – 11
average byte size of active extents, 17 – 6
average size of truncated space, 17 – 6
bringing
online, 17 – 11
online automatically, 17 – 12
online when new, 17 – 8
PARTLY AVAILABLE segment online, 17 – 11
checking if offline, 17 – 12
choosing how many, 2 – 15
choosing size for, 2 – 15
creating, 17 – 8
creating after database creation, 17 – 3
creating public and private, 17 – 3
decreasing size of, 17 – 10
deferred, 17 – 16
displaying
all deferred rollback segments, 17 – 16
deferred rollback segments, 17 – 16
information on, 17 – 15
PENDING OFFLINE segments, 17 – 16
displaying names of all, 17 – 15
dropping, 17 – 14
equally sized extents, 17 – 5
explicitly assigning transactions to, 17 – 13
extends, 17 – 6
guidelines for managing, 17 – 2
initial, 17 – 2
invalid status, 17 – 14
listing extents in, 16 – 26
location of, 17 – 7
making available for use, 17 – 10
managing, 17 – 1
MINEXTENTS and, 10 – 9
monitoring, 17 – 6, 17 – 15
OFFLINE, 17 – 11
offline rollback segments, 17 – 10
offline status, 17 – 12
online rollback segments, 17 – 10
online status, 17 – 12
PARTLY AVAILABLE, 17 – 11
PENDING OFFLINE, 17 – 12
privileges
for dropping, 17 – 14
required to alter, 17 – 10
required to create, 17 – 8
taking online and offline, 17 – 10
setting size of, 17 – 4
shrinks, 17 – 6
status for dropping, 17 – 14
status or state, 17 – 11
storage parameters, 17 – 9
storage parameters and, 17 – 8
SYSTEM rollback segment, 17 – 3
taking offline, 17 – 12
taking tablespaces offline and, 8 – 9
transactions and, 17 – 13
using multiple, 17 – 2
wraps, 17 – 6

ROLLBACK_SEGMENTS parameter
adding rollback segments to, 17 – 8
setting before database creation, 2 – 12

ROWID datatype, 10 – 18

ROWID pseudo–column, 10 – 18
changes in, 10 – 19

rows
chaining across blocks, 10 – 4, 16 – 8
estimating size, A – 4
ROWID changes, 10 – 19
violating integrity constraints, 16 – 13

S

schema objects
auditing, 21 – 11
creating multiple objects, 16 – 2
default audit options, 21 – 13
dependencies between, 16 – 18
disabling audit options, 21 – 14

enabling audit options on, 21 – 13
listing by type, 16 – 25
listing information, 16 – 23
name resolution, 16 – 20
privileges to access, 20 – 6
privileges to rename, 16 – 3
renaming, 16 – 3

SCN. *See* System Change Number (SCN)

security
accessing a database, 18 – 2
administrator of, 18 – 2
application developers and, 18 – 8
auditing policies, 18 – 11
authentication of users, 18 – 2
data, 18 – 3
database administrator's r esponsibilities, 1 – 4
database security, 18 – 2
database users and, 18 – 2
establishing policies, 18 – 1
general users, 18 – 4
multi–byte characters
in role names, 20 – 8
in role passwords, 20 – 10
in user names, 19 – 9
in user passwords, 19 – 10
operating–system role management, 20 – 21
operating–system security and the database, 18 – 3
policies for database administrators, 18 – 7
privilege management policies, 18 – 5
privileges, 18 – 2
protecting the audit trail, 21 – 18
REMOTE_OS_ROLES parameter, 20 – 21
roles to force security, 18 – 5
security officer, 1 – 3
sensitivity, 18 – 3

segments
data and index
default storage parameters, 10 – 11
reducing fragmentation, 10 – 9
data dictionary, 16 – 21
displaying information on, 16 – 26
monitoring, 17 – 15
rollback, 17 – 1
temporary storage parameters, 10 – 12

sensitivity, security, 18 – 3

SEQUENCE_CACHE_ENTRIES parameter, 12 – 11

sequences
altering, 12 – 11
creating, 12 – 10
dropping, 12 – 11
initialization parameters, 12 – 11
managing, 12 – 10
Parallel Server and, 12 – 10
privileges for altering, 12 – 10
privileges for creating, 12 – 10
privileges for dropping, 12 – 11

Server Manager
about, 1 – 16
operating system account, 1 – 4
starting, 2 – 6

server units, composite limits and, 19 – 17

servers
dedicated, multi–threaded contrasted with, 4 – 3
multi–threaded, dedicated contrasted with, 4 – 3

service name, for dispatcher in multi–threaded server, 4 – 6

session limits, license, setting initially, 2 – 12

session monitor, 4 – 10

session, user
active, 4 – 17
inactive, 4 – 18
marked to be terminated, 4 – 17
terminating, 4 – 16
viewing terminated sessions, 4 – 18

sessions
auditing connections and disconnections, 21 – 10
limits per instance, 19 – 2
listing privilege domain of, 20 – 24
number of concurrent sessions, 2 – 12
Parallel Server session limits, 2 – 13
setting maximum for instance, 19 – 3
setting warning limit for instance, 19 – 4
viewing current number and high water mark, 19 – 5
viewing memory use, 19 – 23

SET ROLE command
  how password is set, 20 – 10
  when using operating–system roles, 20 – 21
Set Rollback Segment Offline dialog, 17 – 12
Set Rollback Segment Online dialog, 17 – 11
Set Rollback Segment Storage dialog, 17 – 8
Set Tablespace Offline dialog, 23 – 14
Set Tablespace Online dialog, 8 – 7, 23 – 14
Set Tablespace Storage dialog box, 8 – 5
SET TRANSACTION command, USE ROLLBACK SEGMENT option, 17 – 13
SGA, determining buffers in cache, 2 – 11
shared mode, rollback segments and, 17 – 3
shared pool
  ANALYZE command and, 16 – 7
  multi–threaded server and, 4 – 5
shared server processes
  changing the minimum number of, 4 – 8
  maximum number of, 4 – 8
  number to start initially, 4 – 7
  privileges to change number of, 4 – 8
  trace files for, 4 – 12
shared SQL areas, ANALYZE command and, 16 – 7
shortcuts
  CONNECT, for auditing, 21 – 10
  object auditing, 21 – 11
  object privileges, 20 – 7
  statement level auditing options, 21 – 9
shrunken extents, 17 – 6
Shut Down menu, 3 – 8
  Abort Instance option, 3 – 10
  Immediate option, 3 – 9
  Normal option, 3 – 8
SHUTDOWN command, 3 – 8
  ABORT option, 3 – 10
  IMMEDIATE option, 3 – 9
  NORMAL option, 3 – 8
shutting down a database, 3 – 1
shutting down an instance
  aborting the instance, 3 – 9
  connecting and, 3 – 8
  connecting as INTERNAL, 3 – 8
  example of, 3 – 8
  general procedures, 3 – 8
  immediately, 3 – 9
  multi–threaded server, 3 – 8
  normally, 3 – 8
size
  clusters, A – 9 to A – 15
  datafile, 9 – 3
  hash clusters, 15 – 4
  indexes, A – 5
  on non–clustered tables, A – 2
  rollback segments, 17 – 4
snapshot logs, storage parameters, 10 – 11
snapshots
  media recovery and, 24 – 6
  storage parameters, 10 – 11
  too old, OPTIMAL storage parameter and, 17 – 6
SNP background processes
  about, 7 – 2
  initialization parameters, 7 – 3
  job queues and, 7 – 3
  multiple, 7 – 2
software versions, 1 – 20
SORT_AREA_SIZE parameter, index creation and, 13 – 3
space
  adding to the database, 8 – 3
  used by indexes, 13 – 9
space management
  PCTFREE, 10 – 2
  PCTUSED, 10 – 4
SQL statements
  disabling audit options, 21 – 14
  enabling audit options on, 21 – 12
  privileges required for, 20 – 7
SQL trace facility, when to enable, 4 – 15
SQL*Loader
  about, 1 – 17
  indexes and, 13 – 3
SQL_TRACE parameter, trace files and, 4 – 12
STALE status, of redo log members, 5 – 10
standby database, 24–38
Start Automatic Archiving dialog, 22 – 13
Start Up Instance dialog box, 3 – 2
  Force check box, 3 – 5
  Mount radio button, 3 – 4

  Nomount radio button, 3 – 3
  Open radio button, 3 – 4
  Recover checkbox, 24 – 11
  Restrict to DBAs check box, 3 – 4
  specifying a parameter file, 3 – 3
starting a database
  about, 3 – 1
  general procedures, 3 – 2
  recovering during, 24 – 11
starting an instance
  at database creation, 3 – 3
  automatically at system startup, 3 – 5
  connecting as INTERNAL, 3 – 2
  database closed and mounted, 3 – 3
  database name conflicts and, 2 – 9
  dispatcher processes and, 4 – 6
  enabling automatic archiving, 22 – 6
  examples of, 3 – 5
  exclusive mode, 3 – 5
  forcing, 3 – 5
  general procedures, 3 – 2
  mounting and opening the database, 3 – 4
  multi–threaded server and, 3 – 2
  normally, 3 – 4
  parallel mode, 3 – 5
  parameter files, 3 – 3
  problems encountered while, 3 – 5
  recovery and, 3 – 5
  remote instance startup, 3 – 5
  restricted mode, 3 – 4
  specifying database name, 3 – 2
  troubleshooting, 3 – 3
  with multi–threaded servers, 4 – 4
  without mounting a database, 3 – 3
starting Server Manager, 2 – 6
STARTUP command, 3 – 2
  FORCE option, 3 – 5
  MOUNT option, 3 – 4
  NOMOUNT option, 3 – 3
  OPEN option, 3 – 4
  RECOVER option, 3 – 5, 24 – 11
  RESTRICT option, 3 – 4
  specifying database name, 3 – 2
  specifying parameter file, 3 – 3
statistics, updating, 16 – 4
Stop Auto Archive menu option, 22 – 7
storage
  altering tablespaces, 8 – 5
  quotas and, 19 – 11
  revoking tablespaces and, 19 – 11
  unlimited quotas, 19 – 11
storage parameters
  applicable objects, 10 – 7
  changing settings, 10 – 11
  data dictionary, 16 – 21
  default, 10 – 7
  for the data dictionary, 16 – 21
  INITIAL, 10 – 8, 11 – 7
  INITRANS, 10 – 10, 11 – 7
  MAXEXTENTS, 10 – 8
  MAXTRANS, 10 – 10, 11 – 7
  MINEXTENTS, 10 – 9, 11 – 7
  NEXT, 10 – 8
  OPTIMAL (in rollback segments), 17 – 6
  PCTFREE, 11 – 7
  PCTINCREASE, 10 – 9
  PCTUSED, 11 – 7
  precedence of, 10 – 12
  rollback segments, 17 – 9
  SYSTEM rollback segment, 17 – 9
  temporary segments, 10 – 12
stored procedures
  privileges for recompiling, 16 – 19
  using privileges granted to PUBLIC, 20 – 18
stream, tape drive, 22 – 10
synonyms
  creating, 12 – 12
  displaying dependencies of, 16 – 26
  dropped tables and, 11 – 9
  dropping, 12 – 12
  managing, 12 – 12
  private, 12 – 12
  privileges for creating, 12 – 12
  privileges for dropping, 12 – 12
  public, 12 – 12
SYS
  initial password, 1 – 4
  objects owned, 1 – 5
  policies for protecting, 18 – 7
  privileges, 1 – 5
  user, 1 – 5

SYS.AUD$
  audit trail, 21 – 2
  creating and deleting, 21 – 4
SYSOPER / SYSDBA privileges
  adding users to the password file, 1 – 12
  connecting with, 1 – 14
  determining who has privileges, 1 – 13
  granting and revoking, 1 – 13
SYSTEM
  initial password, 1 – 4
  objects owned, 1 – 5
  policies for protecting, 18 – 7
  user, 1 – 5
System Change Number (SCN)
  checking for a datafile, 9 – 12
  use in distributed recovery, 24 – 6
System Global Area. *See* SGA
System Global Area (SGA), 2 – 11
system identifier, 1 – 10
system privileges, 20 – 2
SYSTEM rollback segment
  adding, 17 – 3
  altering storage parameters of, 17 – 9
  remaining online always, 17 – 11
  rollback segments, 17 – 3
SYSTEM tablespace
  backing up, 23 – 14
  cannot drop, 8 – 12
  initial rollback segment, 17 – 2
  non–data dictionary tables and, 11 – 3
  recovery of, 24 – 20, 24 – 22
  restrictions on taking offline, 9 – 6
  when created, 8 – 3
system time, changing, 24 – 24

T

tables
  allocating extents, 11 – 8
  altering, 11 – 7, 11 – 8
  analyzing statistics, 16 – 3
  clustered, 14 – 2
  clustered tables
    altering, 14 – 9
    creating, 14 – 6
    dropping, 14 – 9
    managing, 14 – 1
    privileges to drop, 14 – 10
    storage, A – 13
  creating, 11 – 6
  designing before creating, 11 – 2
  dropping, 11 – 9
  estimating initial size, A – 2
  estimating size, 11 – 4, A – 4
  guidelines for managing, 11 – 1, 11 – 5
  hash clustered
    creating, 15 – 5
    managing, 15 – 1
  increasing column length, 11 – 7
  indexes and, 13 – 2
  key–preserved, 12 – 5
  limiting indexes on, 13 – 3
  location, 11 – 6
  location of, 11 – 3
  managing, 11 – 1
  parallelizing creation of, 11 – 3
  privileges for creation, 11 – 6
  privileges for dropping, 11 – 9
  privileges to alter, 11 – 7
  schema of clustered, 14 – 7
  separating from indexes, 11 – 5
  specifying PCTFREE for, 10 – 4
  specifying tablespace, 11 – 3, 11 – 6
  storage parameters, 10 – 11
  SYSTEM tablespace and, 11 – 3
  temporary space and, 11 – 5
  transaction parameters, 11 – 3
  truncating, 16 – 9
  UNRECOVERABLE, 11 – 4
  validating structure, 16 – 7
tablespaces
  adding datafiles, 9 – 4
  altering availability, 8 – 7
  altering storage settings, 8 – 5
  assigning defaults for users, 19 – 10
  assigning user quotas, 8 – 3
  backing up offline, 23 – 14
  backing up online, 23 – 10
  backing up several online, 23 – 13
  bringing online, 8 – 7
  checking default storage parameters, 8 – 13
  coalescing, 8 – 6
  creating, 8 – 3
  creating additional, 8 – 3

default quota, 19 – 11
default storage parameters for, 10 – 11
default temporary, 19 – 10
dropping
about, 8 – 12
required privileges, 8 – 12
frequency of backups, 23 – 3
guidelines for managing, 8 – 2 to 8 – 3
listing files of, 8 – 14
listing free space in, 8 – 14
location, 9 – 3
managing, 9 – 1
monitoring, 9 – 12
privileges for creating, 8 – 4
privileges to take offline, 8 – 8
quotas, assigning, 8 – 3
quotas for users, 19 – 11
read–only, 8 – 9
revoking from users, 19 – 11
rollback segments required, 8 – 4
setting default storage parameters for, 8 – 2
SYSTEM tablespace, 8 – 3
taking offline immediately, 8 – 9
taking offline normal, 8 – 8
taking offline temporarily, 8 – 8
temporary, 19 – 10
unlimited quotas, 19 – 11
using multiple, 8 – 2
viewing quotas, 19 – 21, 19 – 22
writeable, 8 – 11

taking offline, tablespaces, 8 – 8

tape drives, streaming for archiving, 22 – 10

temporary segments, index creation and, 13 – 3

temporary space, allocating, 11 – 5

terminating, a user session, 4 – 16

terminating sessions
active sessions, 4 – 17
identifying sessions, 4 – 17
inactive session, example, 4 – 18
inactive sessions, 4 – 18

test, security for databases, 18 – 9

time–based recovery
coordinated in a distributed database, 24 – 5
procedure for, 24 – 24

tip
executing OS commands within Server Manager, 5 – 7

object privilege shortcut, 20 – 7
shortcuts for auditing objects, 21 – 11
statement auditing shortcut, 21 – 9

trace files
control file backups to, 23 – 15
job failures and, 7 – 9
location of, 4 – 14
log writer, 4 – 13
size of, 4 – 14
using, 4 – 12, 4 – 13
when written, 4 – 14

trailing nulls, A – 9

transaction entries, guidelines for storage, 10 – 10

transactions
assigning to specific rollback segment, 17 – 13
rollback segments and, 17 – 13

TRANSACTIONS parameter, using, 17 – 2

TRANSACTIONS_PER_ROLLBACK_ SEGMENT parameter, using, 17 – 2

triggers
auditing, 21 – 22
disabling, 16 – 11
dropped tables and, 11 – 9
enabling, 16 – 11
examples, 21 – 23
privileges for controlling, 20 – 7
privileges for enabling and disabling, 16 – 11

TRUNCATE command, 16 – 9
DROP STORAGE option, 16 – 10
REUSE STORAGE option, 16 – 10

truncated extents, 17 – 6

truncating
clusters, 16 – 9
privileges for, 16 – 10
tables, 16 – 9

Trusted Oracle
role management, 20 – 1
security policies for, 18 – 1
special datatypes, 10 – 19
system privileges, 20 – 1

Trusted Oracle7 Server
auditing, 21 – 1
controlling database access, 19 – 1
database migration to, 2 – 3

LOG_CHECKPOINT_INTERVAL and, 5 – 12
managing rollback segments, 17 – 1
managing tablespaces and datafiles, 9 – 1
managing users and resources, 19 – 1
shutting down a database, 3 – 8
tuning
archiving, 22 – 9
databases, 1 – 20
initially, 2 – 14
tuning parameters, 2 – 14

U

UNIQUE key constraints
disabling, 16 – 14
dropping associated indexes, 13 – 10
enabling, 16 – 14
enabling on creation, 13 – 6
foreign key references when dropped, 16 – 15
indexes associated with, 13 – 6
storage of associated indexes, 13 – 7
UNLIMITED TABLESPACE privilege, 19 – 12
unrecoverable
objects and recovery, 24 – 46
tables, 11 – 4
unrecoverable indexes, indexes, 13 – 5
UPDATE privilege, revoking, 20 – 16
USER_DUMP_DEST parameter, 4 – 14
USER_EXTENTS, 9 – 12
USER_FREE, 8 – 13, 9 – 12
USER_INDEXES view, filling with data, 16 – 5
USER_SEGMENTS, 8 – 13, 9 – 12
USER_TAB_COLUMNS view, filling with data, 16 – 5
USER_TABLES view, filling with data, 16 – 5
USER_TABLESPACES, 8 – 13, 9 – 12
usernames, SYS and SYSTEM, 1 – 4
users
altering, 19 – 12
assigning profiles to, 19 – 17
assigning tablespace quotas, 8 – 3
assigning unlimited quotas for, 19 – 11
authentication, database authentication, 19 – 8
authentication
about, 18 – 2, 19 – 6
operating–system authentication, 19 – 7
changing authentication method, 19 – 13
changing default roles, 19 – 13
changing passwords, 19 – 13
composite limits and, 19 – 17
default tablespaces, 19 – 10
dropping, 19 – 15
dropping profiles and, 19 – 19
dropping roles and, 20 – 11
end–user security policies, 18 – 5
enrolling, 1 – 20
identification, 19 – 6
in a newly created database, 2 – 14
limiting number of, 2 – 13
listing, 19 – 20
listing privileges granted to, 20 – 23
listing roles granted to, 20 – 24
managing, 19 – 8
multi–byte characters
in names, 19 – 9
in passwords, 19 – 10
objects after dropping, 19 – 15
password security, 18 – 4
policies for managing privileges, 18 – 5
privileges for changing passwords, 19 – 12
privileges for creating, 19 – 9
privileges for dropping, 19 – 15
PUBLIC group, 20 – 18
security and, 18 – 2
security for general users, 18 – 4
session, terminating, 4 – 18
specifying user names, 19 – 9
tablespace quotas, 19 – 11
unique user names, 2 – 13, 19 – 5
viewing information on, 19 – 21
viewing memory use, 19 – 23
viewing tablespace quotas, 19 – 21, 19 – 22
utilities
Export, 1 – 17
for the database administrator, 1 – 16
Import, 1 – 17
Server Manager, 1 – 16
SQL*Loader, 1 – 17
UTLCHAIN.SQL, 16 – 8
UTLEXCPT.SQL, running, 16 – 16

UTLLOCKT.SQL script, 4 – 10

V

V$ARCHIVE view, 22 – 10
V$BACKUP view, 23 – 12
V$DATABASE view, 22 – 10
V$DATAFILE, 8 – 13, 9 – 12
V$DATAFILE view, 23 – 8
V$DBFILE view, 2 – 8
V$DISPATCHER view, controlling dispatcher process load, 4 – 9
V$LICENSE view, 19 – 5
V$LOG view
  displaying archiving status, 22 – 10
  online redo log and, 5 – 15
V$LOGFILE view, 2 – 8
  listing files before backup, 23 – 8
V$PWFILE_USERS view, 1 – 13
V$QUEUE view, controlling dispatcher process load, 4 – 9
V$RECOVERY_FILE view, 24 – 8
V$RECOVERY_FILE_STATUS, 24 – 14
V$RECOVERY_STATUS, 24 – 14
V$ROLLNAME, finding PENDING OFFLINE segments, 17 – 16
V$ROLLSTAT, finding PENDING OFFLINE segments, 17 – 16
V$SESSION, 7 – 13
V$SESSION view, 4 – 18
V$THREAD view, 5 – 15
VALIDATE STRUCTURE option, 16 – 7
VARCHAR2 datatype, 10 – 17
  space use of, 10 – 17
versions
  See also releases
  of other Oracle software, 1 – 21
views
  creating, 12 – 2
  creating with errors, 12 – 4
  displaying dependencies of, 16 – 26
  dropped tables and, 11 – 9
  dropping, 12 – 9
  FOR UPDATE clause and, 12 – 3
  managing, 12 – 1, 12 – 9
  ORDER BY clause and, 12 – 3
  privileges, 12 – 2
  privileges for dropping, 12 – 9
  privileges for recompiling, 16 – 19
  privileges to replace, 12 – 8
  recompiling, 16 – 19
  replacing, 12 – 8
  wildcards in, 12 – 3
  WITH CHECK OPTION, 12 – 3
violating integrity constraints, 16 – 13

W

warning
  archiving mode for first backup, 23 – 6
  changing data dictionary storage parameters, 16 – 21
  consistency and Export backups, 23 – 18
  creating a rollback segment, 2 – 12
  data in dropped tablespace is unrecoverable, 8 – 12
  datafile names missing from control file, 23 – 6
  disabling audit options, 21 – 14
  enabling auditing, 21 – 12
  keeping symmetric online redo log groups, 5 – 2
  number of control files, 6 – 2
  setting the CONTROL_FILES parameter, 2 – 10
  use mirrored control files, 6 – 2
wildcards, in views, 12 – 3
WORM devices, and read–only tablespaces, 8 – 11
writeable tablespaces, 8 – 11

Reader’s Comment Form

Name of Document: Oracle7™ Server Administrator’s Guide
Part No. A32535–1

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the topic, chapter, and page number below:

Please send your comments to:

Oracle7 Server Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood City, CA 94065 U.S.A.
Fax: (415) 506–7200

If you would like a reply, please give your name, address, and telephone number below:

Thank you for helping us improve our documentation.